uc3m | Universidad Carlos III de Madrid

Grado Ingeniería en Sistemas Audiovisuales

2018 - 2019

Trabajo Fin de Grado

"Diseño e implementación de un módulo Python para representar datos geográficos en Dataframes"

Isabela San José García

Tutor/es:

Antonio Ángel Pastor Valles

Leganés, junio 2019

A mis padres, que lo dan todo. Por cada día que me han empujado y animado para que sea posible. Y a mi hermano Gonzalo, que siempre está para apoyarme.

A mis 5: Pat, Lara, Mary, Nair y Kalamar. Por la paciencia, los ánimos y el cariño.

A los chicos del Maccabi: Palen, Mike y Rubi. Junto con Coca y JC. Que siempre saben qué decir.

A los que me han llevado de la mano en este viaje: Chuchis, Pol, Lauri, Jano, Aitor, Viki, Victor, Julio y mi querida Irina. Por sufrir juntos, y por compartir.

Nombramiento especial a dos personas que han dado mucho de su tiempo y dedicación:

Mi buen colega Maxi y mi chico, Luis (por cada día a mi lado). Sin ellos esto sería un folio en blanco.

Y a mi tutor, por la inspiración, gracias.

“You don't pay the plumber for banging on the pipes. You pay him for knowing where to bang.”

RESUMEN

Es extremadamente difícil obtener información útil de datos espaciales utilizando tablas como forma de representación, en este proyecto se han combinado estas tareas. Por lo que a partir de tablas, usando Dataframes de entrada (con una cantidad significante de datos), se ha encontrado la manera de representar sus valores mediante datos espaciales en un mapa (utilizando la librería Folium de Python).

Este proyecto consiste en crear un módulo en Python para la representación geográfica de datos de manera visual. Además de ser código abierto, destaca la facilidad de su uso.

Durante el desarrollo del módulo se fusionan dos soluciones que plantea Python, el análisis espacial y la gestión de grandes volúmenes de datos, por lo que puede ser empleado en empresas de Big Data.

Este módulo trata de crear una librería Python la cual de manera sencilla es capaz de transformar un conjunto grande de datos de entrada en representaciones geográficas de dichos datos. Esta librería engloba unas funciones concretas que tienen la capacidad de ser adaptables a la representación de estadísticas y estudios geográficos. Las funciones de las que consta la librería descrita son:

- Representaciones de Dataframes como puntos de interés.
- Representación de Dataframes en mapas coropléticos.
- Representación de Dataframes con URLs de Google Maps.
- Representaciones de Dataframes en Geohash.

Python se introdujo en el mundo de los sistemas de información geográfica (SIG) como un lenguaje de programación relativamente fácil de aprender y de utilizar. Este lenguaje con el paso del tiempo y por su clara expansión, se ha vuelto omnipresente, ofreciendo soluciones para distintos tipos de usuarios.

Todo esto, se realiza a través de la creación de un módulo Python que lo combinará en un solo bloque, transportable a cualquier navegador y a cualquier dispositivo móvil. Este módulo está completamente basado en varias librerías muy importantes en Python: Numpy, Pandas y Folium.

Principalmente Leaflet es la base de este proyecto, está basada en una librería JavaScript que nos permite publicar mapas, estadísticas y gráficos en la web de forma rápida y eficaz. Mediante Folium se ha conseguido unir ambos mundos, para manipular datos en Python y visualizar grandes volúmenes de datos para una representación SIG.

Visualizar datos espaciales utilizando mapas, tiene sus ventajas ya que se puede obtener una representación visual de la ubicación exacta de los puntos, conjunto de datos geográficos, esto nos permite relacionar fácilmente los puntos que tenemos con el mundo real. También nos permite generar perspectivas geográficas a partir del conjunto de datos y posiciones que tenemos.

El trazado de datos espaciales en un mapa nos permite obtener información geográfica de un modo que no podremos obtener con otras formas de representación, ni mediante otros tipos de gráficos.

Por eso, el uso de mapas en lugar de otras formas de gráficos nos permite resaltar tendencias, estadísticas, descubrir patrones y revelar realidades no visibles con representaciones de datos espaciales. Enseñándonos una claridad sobre los datos.

Como declaró Alberto Cairo en su libro “El arte funcional: una introducción a los gráficos y la visualización de la información”:

“Los gráficos no deben simplificar los mensajes. Deben aclararlos, resaltar tendencias, descubrir patrones y revelar realidades que antes no se veían”

Palabras clave:

Mapas coropléticos; Geohashes; Folium; Librería Python; Pandas;

CONTENIDO

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Motivación

En el campo de *Data Science*[1], y a la hora mostrar los resultados obtenidos, es muy importante su representación gráfica, para poder transmitir de una manera sencilla de entender los resultados del estudio, ya sea destinado a otros investigadores, compañeros o a personas interesadas en nuestra publicación.

Es muy importante la manera de transmitir la información de un modo visual, puesto que se debe de complementar en conjunto con nuestros resultados analíticos. Mediante representaciones se pueden observar patrones, diferencias, estadísticas, agrupaciones o flujos de forma más visual e instantánea que si se representaran los resultados en tablas o de manera estática. Siempre teniendo en cuenta que esta no es una forma de simplificar el resultado, si no de resaltarlo y hacerlo más sencillo de observar e interpretar.

Hacer una buena visualización de datos no es un problema sencillo, existen muchos estudios al respecto que estudian la percepción humana y cómo nuestro cerebro interpreta la información, por eso, siempre es muy importante que se transmita correctamente el mensaje que se desea.

Al comenzar este proyecto, al igual que en algunos trabajos realizados durante la carrera, he observado que existe una carencia común en todos los casos de la representación visual de datos. En la mayoría de los casos se observan representaciones que no llegan a alcanzar el mayor potencial que podrían adquirir, de ser una buena visualización, y en otros casos, la representación es muy vaga o muy poco atractiva, por lo tanto, no llama la atención.

Existen infinidad de librerías para hacer dicha visualización, pero son muy complicadas de utilizar para llegar a lograr un buen resultado, y normalmente, requieren mucho tiempo. Por ello, estoy interesada en poder proveer de una herramienta que, de manera clara y concisa, muestre a los científicos de datos e investigadores una buena visualización de los resultados, sin que ello requiera que tengan que adquirir unas altas capacidades de programación, y obteniendo resultados en poco tiempo. De esta manera, poder potenciar sus gráficos y mejorar la transmisión de la información a los lectores.

[1] Ciencia de los datos

Además de esta motivación científica, personalmente, este trabajo es un reflejo de mi interés en la investigación, puesto que entre todos los campos abarcados durante mi carrera siempre me ha llamado la atención este campo del *Data Science.*

Durante la carrera he tenido varias asignaturas relacionadas con la programación, y especialmente con C, Java y Matlab. Esto ha hecho que me interese bastante por la programación enfocada a los estudios estadísticos y a la representación más visual de los datos. Puesto que tenía algo de base programando, decidí aprender un lenguaje nuevo, como es el de Python, ya que gracias a que está teniendo una explosión de su uso junto con la utilización de los datos en las empresas tiene un gran crecimiento profesional, y esto me parecía que podía ayudarme en el futuro.

1.2. Área de la aplicación

Entre los denominados “lenguajes de la ciencia de datos”, Python es uno de los más extendidos que se utiliza habitualmente para realizar cálculo estadístico, así como la visualización gráfica de datos, extraer conclusiones y poder llegar a tomar decisiones finales de estudios e investigaciones. También hay una gran comunidad alrededor, que provee de herramientas y librerías destinada a la *minería de datos, Big data, inteligencia artificial* y algunos otros campos.

Resulta un lenguaje muy interesante no solo para visualizar datos, sino por su capacidad para extraerlos, para automatizar algunos procesos o para utilizarlo en el aprendizaje automático. Con Python se pueden cambiar, modificar o extraer grandes grupos de datos interesantes, mediante la utilización de librerías o módulos, como son el de Numpy, Pandas, Folium, Json, ...

Y aunque su uso está destinado para casi todos los campos que engloban la ciencia de datos, las librerías destinadas a la visualización de dichos datos son uno de los puntos más importantes. Su número no para de crecer, ya que se pretende que cada una se adapte mejor a los datos de entrada que poseen, o al tipo de salida que se desea.

1.3. Sistema de Información Geográfica (SIG)

El sistema de información geográfica (SIG) es un gestor de integración que combina software con datos geográficos para diseñar, capturar, almacenar, manipular, analizar y desplegar de cualquier manera posible (dentro de sus limitaciones) la información geográficamente referenciada con el fin de llegar a resolver problemas más complejos de gestión y planificación.

Su funcionamiento principal es similar al de una base de datos que contiene información geográfica (datos alfanuméricos), esta información se encuentra asociada a través de un identificador común a todos los objetos que pertenecen a un mapa digital. De manera que señalando un único objeto se pueden conocer todos sus atributos, así como de manera contraria, preguntando por uno de los registros de esta base de datos se podría conseguir su localización dentro de un conjunto de datos geográficos.

Un SIG se utiliza principalmente para gestionar la información espacial, ya que este sistema nos permite segmentar toda la información que pertenece a los atributos de los objetos en diferentes capas temáticas, y de esta manera, almacenarlas independientemente. Estas capas temáticas se pueden separar, permitiendo trabajar con ellas de forma sencilla y rápida, con el fin de obtener otra nueva capa, que no podríamos generar de otra manera.

Las cuestiones principales que se pueden resolver mediante un SIG son:

- Localización: Permitiendo preguntar por las características pertenecientes a un lugar en concreto.
- Condición: El cumplimiento de varias condiciones y funciones impuestas al sistema.
- Tendencia: Comparaciones entre situaciones espaciales y temporales distintas de alguno de los atributos.
- Rutas: Cálculo de rutas óptimas entre puntos.
- Pautas: Detección de pautas espaciales especificadas.
- Modelos: La generación de modelos partiendo de fenómenos o sucesos simulados.

Como observamos, el campo de aplicación de los SIG es muy numeroso, por lo que se podría utilizar en multitud de actividades, siempre y cuando tenga un componente espacial. Son las nuevas tecnologías las que han impulsado a los SIG a su decisivo desarrollo, evolución e implementación

2. ESTADO DEL ARTE

Las gráficas y las representaciones pueden proporcionarnos mucha información de un solo vistazo, pero no siempre son el mejor de los métodos de representación. A veces es necesario ir un poco más lejos y representar valores, puntos de interés y estadísticas a través de un mapa geográfico, el cual nos permite ubicarnos en el área y nos da la información que necesitamos. Si el mapa además tiene la capacidad de interactuar con nosotros, de modo que podamos viajar con él y movernos a otro punto distinto para consultar los mismos valores y compararlos, nos facilitaría mucho la manera de recibir esta información.

Para favorecer las necesidades de muchos científicos y para poder representar datos junto con puntos geográficos y mapas con Python nacieron varias librerías:

PyNGL

También llamado “pingle” es un módulo de Python que genera visualizaciones 2D en alta calidad para datos científicos, especialmente para representaciones del tiempo y clima. Es capaz de generar proyecciones del mapa del mundo con contornos y combinarlos con gráficos en color que muestran las evoluciones climáticas.

Se suele combinar con el módulo PyNIO (“pie nee oh”), que se utiliza para leer y escribir en distintos formatos de datos climáticos.

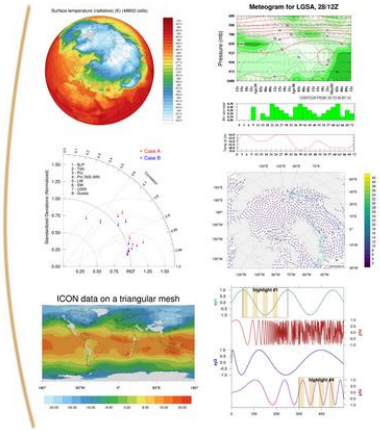

Ilustración 1. Distintos tipos de gráficos y representaciones mediante PyNGL y PyNIO.[2]

[2] Captura de pantalla de https://www.pyngl.ucar.edu/

MATPLOTLIB

Corresponde a un módulo en Python que además de crear gráficos de áreas, histogramas, visualizaciones de líneas, barras y diagramas de dispersión, entre otros, puede hacer visualizaciones con mapas, eso sí, combinándolo con otros módulos como Basemap (para 2D) o Mplot3D (para 3D).

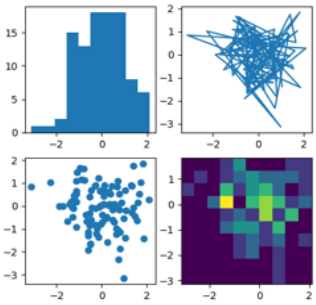

Ilustración 2. Distintas representaciones mediante la librería Matplotlib.[3]

Es la librería más utilizada en Python, para ciencia de datos. Su éxito está basado en la facilidad que da a los desarrolladores, a la hora de diseñar visualizaciones con datos a partir de muy pocas líneas de código, mezclada con la posibilidad de incluir esos gráficos en cualquier proyecto web.

Para la representación de mapas no es la más completa, ya que te da la posibilidad, pero la interacción es limitada. Por ejemplo, te deja hacer zoom ante la propia visualización, pero poco más.

BASEMAP

Basemap es una herramienta excelente para crear mapas usando Python de una manera sencilla y poderosa, al ser una extensión de Matplotlib, por lo que tiene todas sus características de representación de datos integradas, agregando además las proyecciones geográficas básicas, y algunos conjuntos de datos para dibujar líneas costeras y delimitaciones de países, entre otros.

[3] Captura de pantalla de https://matplotlib.org/3.1.0/tutorials/index.html

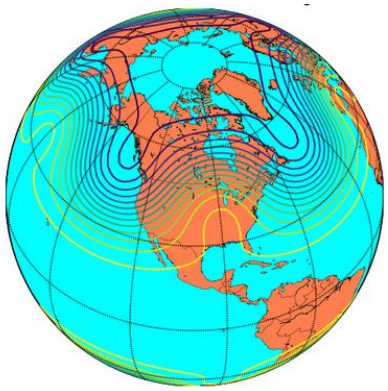

Ilustración 3. Ejemplo de representación Basemap sobre globo terráqueo.

Es la librería más utilizada cuando se utiliza Jupyter con Matplotlib, aunque se pueden representar mapas, su utilización e implementación es bastante complicada.

Existen varias formas de instalación, lo cual hace que sea algo confuso. Dicha instalación depende del método en el que tengas ya instalado Matplotlib.

Esta herramienta tiene alguna documentación, pero existen muchas cosas destinadas a su implementación que son muy difíciles de encontrar. Su documentación y sus ejemplos crecieron hace unos años y cubre muchas posibilidades, pero aun así su información es escasa.

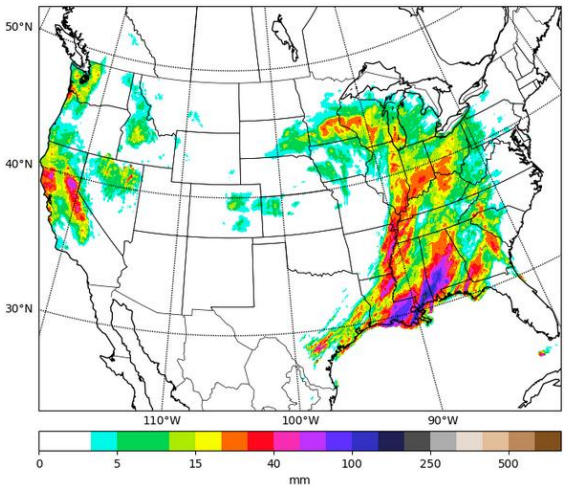

Ilustración 4. Representación de lluvias durante un periodo mediante Basemap.[4]

[4] Capturas de pantalla de https://matplotlib.org/basemap/users/examples.html

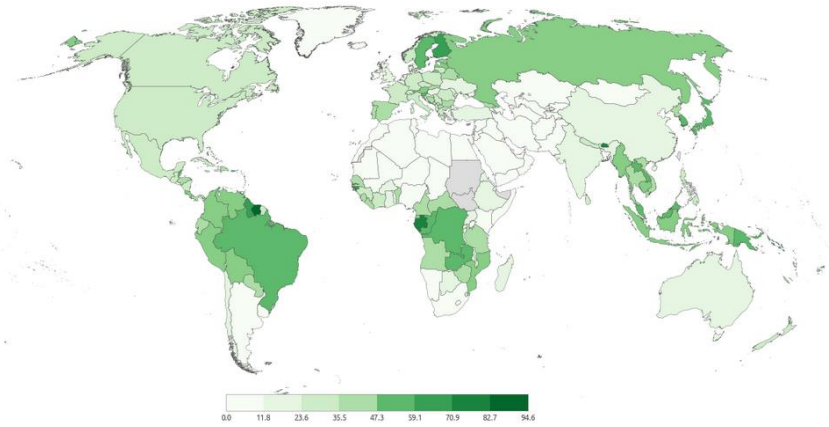

Ilustración 5. Representación del porcentaje de vegetación mediante Basemap.[5]

MPLOT3D

Esta es la librería menos popular, ya que es bastante complicada de usar y algo diferente, el kit de herramientas de Mplot3d agrega capacidades de trazado 3D simples a Matplotlib (ya que trabajan juntos) al proporcionar un objeto de ejes que es capaz de crear una proyección en 2D de una escena 3D.

Aunque este módulo no tenga mucha comunidad, es posible crear mapas con elementos 3D y Basemap, junto con la librería Matplotlib.

Para realizar en Mplot3d un Basemap, se debe usar la clase Axes3D, para agregar datos geográficos al mapa que se desea representar. La variable que se utiliza como instancia de esta clase, debe de ser compatible con las operaciones 3D y esto no ocurren en muchos casos con los métodos que pertenecen a Basemap.

Se tiene que asignar un bloque para que nos diga cómo se debe rotar el mapa resultante, de este modo, la vista es mejor.

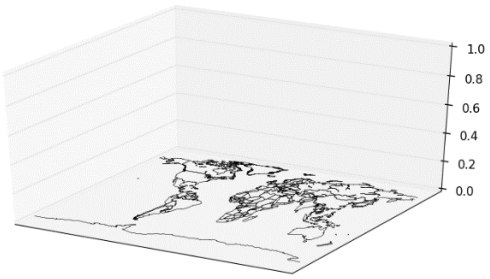

Ilustración 6. Representación en 3D del mapa del mundo mediante la librería Mplot3d.

[5] Captura de pantalla de http://ramiro.org/notebook/basemap-choropleth/

Existe una rotación predeterminada que consigue que se vea en 2D. La rotación del eje se establece para que el mapa se vea desde el eje z, como cuando se dibuja en papel.

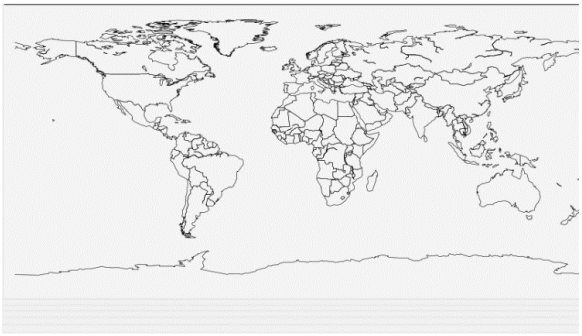

Ilustración 7. Representación en 2D del mapa del mundo mediante la librería Mplot3d.

El objetivo de crear un mapa 3D es acompañarlos de datos y estadísticas 3D en él. Para ello, la clase Axes3D tiene el método bar3d que dibuja diagrama de barras en 3D, y se puede agregar en el mapa creado utilizando la tercera dimensión. El mapa se amplía para adaptarse a las necesidades del conjunto de datos a mostrar, y los ejes se pueden eliminar.

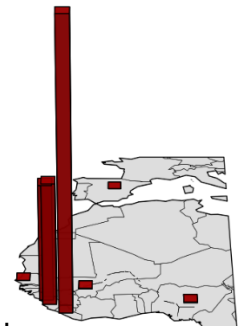

Ilustración 8. Representación de datos en 3D mediante la librería Mplot3d.[6]

Esta librería contiene funciones interactivas que brindan la capacidad de rotar y hacer zoom en el conjunto perteneciente al mapa 3D.

SEABORN

Es una de las librerías de visualización de datos en Python, está basada en Matplotlib. La idea que presenta Seaborn es que los científicos de datos dispongan de una interfaz para hacer gráficos estadísticos atractivos y explicativos.

Seaborn ofrece el objetivo de visualizar datos complejos de una manera sencilla y extraer conclusiones sobre esos datos.

[6] Capturas de pantalla de https://basemaptutorial.readthedocs.io/en/latest/basemap3d.html

Entre algunas de sus características, están las principales:

- Tiene varios temas integrados, que mejoran el diseño de Matplotlib.
- Dispone de herramientas visuales para elegir los colores deseados (paletas de color).
- Posee funciones que compara subconjuntos de datos definidos.
- Se pueden configurar herramientas que sirven para adaptar y visualizar modelos estadísticos.
- Incluye funciones para visualizar matrices de datos.
- Utiliza algoritmos de clustering.

BOKEH

El objetivo final de Bokeh es representar gráficos elegantes, atractivos y sencillos, con el estilo de una librería de JavaScript, proporcionando una interactividad con estos gráficos de alto nivel, ya que puede interactuar con grandes volúmenes de datos. Es un opción muy interesante y válida para representar mapas y otras representaciones si no conoces como funciona Excel, ya que oferta una alternativa a las opciones existentes.

Las representaciones con Bokeh, están orientadas especialmente a navegadores modernos. Siendo capaz de representar mapas coropléticos, mapas de calor, gráficos de línea y áreas.

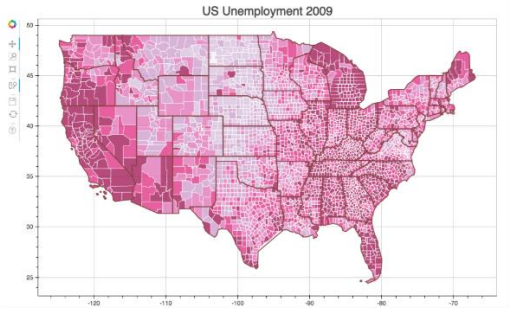

Ilustración 9. Representación mediante Bokeh de la tasa de desempleo en US 2009.

Tienen muchas opciones distintas, en función de los datos a mostrar, Bokeh ofrece una funcionalidad y una visualización apropiada en cada caso.

Bokeh dispone de tutoriales para desarrolladores y científicos de datos, para poder trabajar con esta librería, en estos tutoriales podemos encontrar formación específica para cada tipo de gráfico y formación más avanzada para interactuar con ellos.

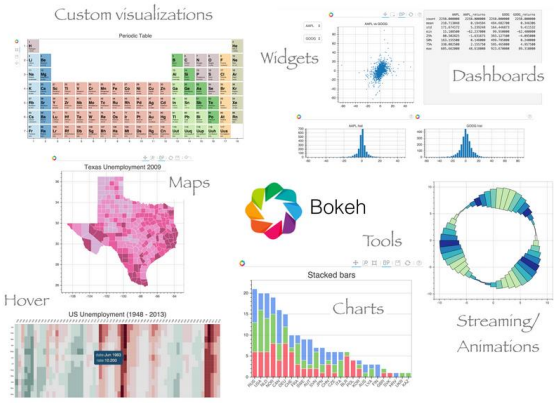

Ilustración 10. Distintas representaciones de estadísticas mediante la librería Bokeh de Python.[7]

PYGAL

Pygal se utiliza fundamentalmente para crear gráficos en formato SVG, esto es algo muy útil y habitual si se desea crear visualizaciones interactivas para distintos tipos de proyectos digitales, ya que se integra fácilmente con Flash.

Con las librerías que nos ofrece Pygal se pueden realizar gráficos de líneas, gráficos de barras, gráficos circulares (de forma de tarta), gráficos de embudo y sirve también para representar todo tipo de visualizaciones con mapas.

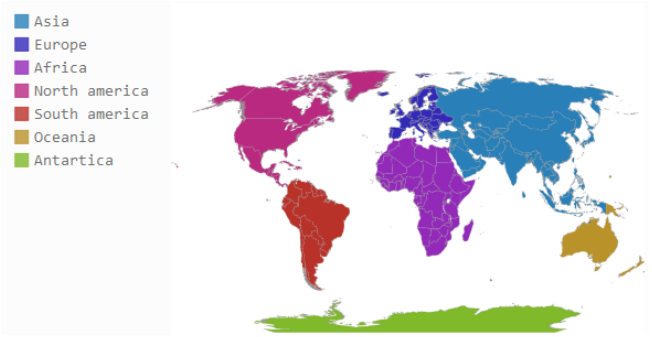

Ilustración 11. Representación de continentes mediante la librería Pygal.[8]

Pygal permite descargar, editar e imprimir todas estas gráficas en formato de imagen, es decir con la extensión .png, y además con muy alta calidad, pero se deben instalar varias dependencias que lo permiten.

En la representación de estadísticas sobre mapas con este módulo, deberíamos tener cuidado con las fechas. Ya que, si se tienen fechas con zonas horarias, nos debemos asegurar de que todas sus fechas tengan una misma zona horaria, de lo contrario, pueden existir bastantes incoherencias a la hora de pintar los mapas.

[7] Capturas de pantalla de http://bokeh.pydata.org/en/0.11.0/docs/gallery/choropleth.html
[8] Captura de pantalla de http://www.pygal.org/en/stable/documentation/types/maps/

PLOTY

Ploty corresponde a una librería de Python algo distinta a las demás, ya que es una librería para el análisis y la visualización de datos de manera online.

Dispone de una documentación bastante completa, que tiene muchos ejemplos y tutoriales muy accesibles, sirve para realizar cualquier tipo de gráficos a partir de diseños basados en Matplotlib y además diseños provistos por la API.

La API nos permite manejar de manera libre datos para realizar gráficas, que posteriormente nos servirán para descargar un formato de imagen o para integrar una aplicación externa.

Para poder utilizar la API en Python hace falta una instalación y seguir los pasos indicados pro la documentación propia.

Cuando se utiliza Ploty se pueden llegar a importar datos desde Google Drive, Dropbox e incluso se pueden combinar con servicios de gestión de bases de datos (MySQL, Oracle, SparkSQL…) para realizar visualizaciones. También nos oferta la posibilidad de descargar datos a Excel.

Mediante Ploty se pueden hacer todo tipo de visualizaciones: gráficos de barras, gráficos de áreas, histogramas, diagramas de burbujas, mapas de calor, mapas de líneas y mapas coropléticos.

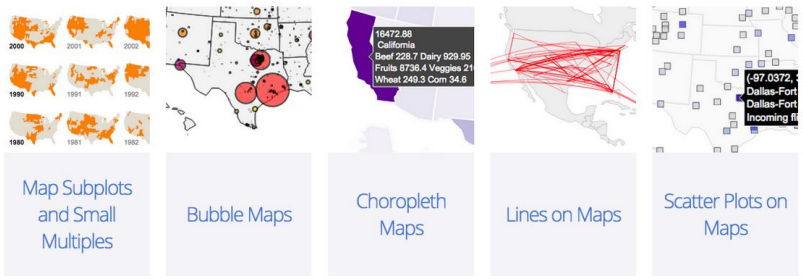

Ilustración 12. Distintos ejemplos de visualizaciones realizadas mediante la librería Ploty.[9]

Los gráficos que se generan pueden ser mostrados de manera alternativa, tanto de forma individual como múltiple, es decir, que pueden generarse varios gráficos y además, mostrarlos en paralelo, ordenarlos en columnas o en filas y combinarse en uno solo, obteniendo de este modo, distintas visualizaciones de datos mediante un solo vistazo.

[9] Captura de pantalla de https://plot.ly/python/maps/

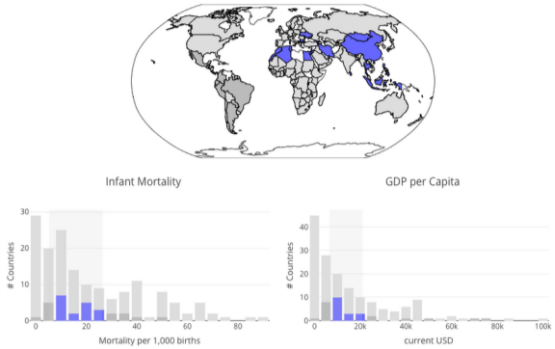

Ilustración 13. Representación combinada de mapas y diagrama de barras, mediante Ploty.[10]

Además, estos gráficos pueden ser exportados en distintos formatos, como .png, PDF, SVG, entre otros, dejando escoger las medidas de alto y ancho para su posterior visualización.

Existen dos entornos para desarrollar en Ploty. Uno es de acceso abierto para los gráficos, siempre y cuando te registres, ya que entonces podrás disponer de un almacenamiento ilimitado para visualizaciones públicas, permitiendo así que otros usuarios que dispongan del mismo servicio puedan realizar comparaciones o utilizar gráficos existentes como ejemplos y para adaptar sus propios datos a estas representaciones. El otro entorno es su versión gratuita, que permite alojar gráficas, pero de manera privada, en este caso existen grandes limitaciones de espacio, e impide que otros usuarios, sea cual sea el tipo de entorno del que dispongan, puedan acceder a los datos privados de los gráficos.

Se puede llegar a configurar Ploty para trabajar en modo sin conexión, pero esto debe de ser implementado únicamente mediante cuadernos[11] de Jupyter.

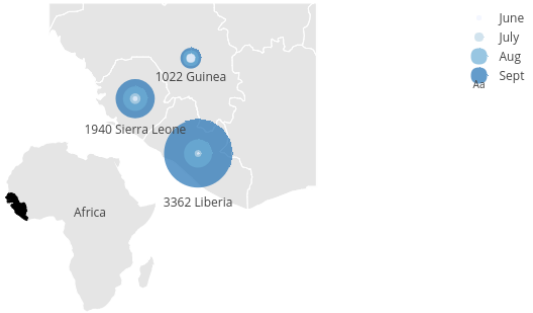

Ilustración 14. Representación de los casos reportados de Ébola en África Occidental mediante Ploty. [12]

[10] Captura de pantalla de http://nicolas.kruchten.com/content/2017/11/plotly_crossfilter/
[11] Cuadernos de Jupyter, denominados *notebooks*.
[12] Captura de pantalla de https://plot.ly/python/choropleth-maps/

IPYLEAFLET

Ipyleaflet es una librería que tiene implementación en Python, se usa de conector con la biblioteca Leaflet.js (biblioteca de JavaScript), que nos permite administrar sus datos fácilmente mediante Python, de este modo se puede realizar un mapa interactivo utilizando el amplio poder que nos concede JavaScript.

Su uso mediante *notebooks* de Jupyter está bastante extendido, ya que nos brinda la posibilidad de realizar una representación interactiva de los datos.

A diferencia de otras librerías, Ipyleaflet es una extensión específica de Jupyter que brinda una integración completa en la herramienta, obteniendo así un mayor grado de sintonía en la interfaz.

Para poder usarlo, se deben instalar las dependencias propias de Ipyleaflet en la misma máquina en la que se pretende ejecutar Jupyter, instalar la extensión en el mismo y reiniciarlo. De esta manera, ya se podrían utilizar todas las funciones que esta extensión nos provee, por ejemplo:

- Leer un GeoJson.
- Poder utilizar las funciones básicas de la biblioteca Leaflet.js
- Integrar controles[13], que permiten cambiar variables en tiempo real.
- Dividir una representación geográfica en varios sectores, mostrando diferentes capas.
- Representar movimiento dentro del mapa, mediante animaciones (viento, tráfico, migraciones, condiciones atmosféricas…)
- Representar mapas coropléticos.

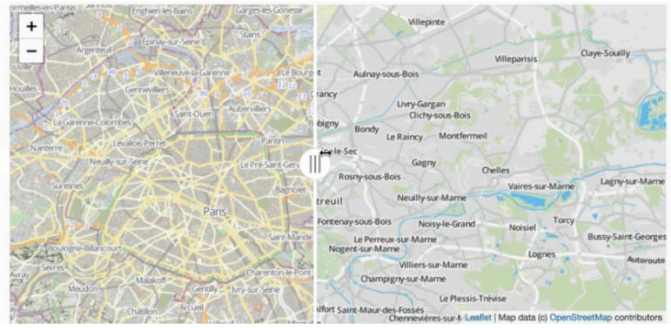

Ilustración 15. División de un mapa en dos sectores para la representación de distintas capas.[14]

[13] *Knobs.* Controles, traducidos como mandos o sliders en Jupyter. Externos a la representación resultante del *notebook*, pero que pertenecen a los propios *notebooks* de Jupyter.

[14] Captura de pantalla de https://github.com/jupyter-widgets/ipyleaflet

Es una librería muy sencilla de utilizar dentro de el entorno de Jupyter, y eso permite hacer pruebas muy rápidas. Tras el uso y estudio minucioso de sus características y limitaciones, podemos concluir que ha sido una librería muy útil para empezar a trabajar y para probar muchas funciones de una manera muy rápida, pudiendo comprobar resultados al mismo tiempo que programamos.

FOLIUM

Folium es una librería bastante completa de Python que incluye la biblioteca Leaflet.js (biblioteca de JavaScript) y que nos permite administrar sus datos fácilmente mediante Python, de este modo podemos realizar un mapa interactivo utilizando el amplio poder que nos concede JavaScript.

Observando la documentación que posee esta librería, nos damos cuenta de que es muy amplia, por lo que nos encontramos numerosos ejemplos para poder practicar y para poder crear mapas coropléticos.

Con el fin de construir un mapa coroplético necesitamos, principalmente, dos elementos:

- Un archivo en formato GeoJson, que nos proporciona los límites de cada zona a representar.
- Un marco de datos, que da los valores de cada zona.

A través de Folium se pueden crear varios tipos de mapas interactivos, por esta razón, esta biblioteca nos es muy útil para la creación de tableros[15]. Pudiendo mostrar muchos datos especiales, es decir, todo tipo de datos que contienen información geográfica (latitud, longitud, altitud) como parte de sus características, además de datos estadísticos, grafos y resúmenes visuales sobre el tipo de datos que, a su vez, se están mostrando en el mapa coroplético representado.

[15] Tablero. *Dashboard*.

Ilustración 16.Representación de las centrales hidroeléctricas de Brasil mediante Folium.[16]

Esta librería se basa en los puntos fuertes de los datos de Python, y en los puntos fuertes de la biblioteca Leaflet.js, de esta manera crean una simbiosis muy completa. La cual usaremos para implementar las funciones que se van a implementar en este proyecto. Nos ofrece la visualización de datos espaciales de manera interactiva, y nos ofrece interacción con Jupyter, además de poder utilizarse independientemente de él. La biblioteca es muy intuitiva de utilizar y ofrece un alto grado de interactividad. Además de tener la gran ventaja de que es *Open Source*[17].

[16] Captura de pantalla de https://ocefpaf.github.io/python4oceanographers/blog/2015/12/14/geopandas_folium/

[17] *Open Source.* Código abierto. Que permite la utilización de este código en otros proyectos, bajo una licencia determinada. La licencia más común es la MIT.

3. IPYLEAFLET VS FOLIUM

Ya que el objetivo de este proyecto es elaborar una buena representación geográfica, hemos aislado en la búsqueda por la mejor librería a dos candidatos para desarrollar con ellos toda la investigación propuesta. Estos dos finalistas son Ipyleaflet y Folium.

Al principio, fue muy costoso encontrar una biblioteca adecuada en el mundillo siempre cambiante de las bibliotecas de Python, pero después de algunas investigaciones descubrí Folium, lo que facilita la creación y manipulación de representaciones de datos geográficos mediante Python.

Para concluir la utilización de Folium, hemos probado en profundidad las librerías mencionadas en el capítulo anterior[18], analizando cuidadosamente sus ventajas y desventajas. Y después de comparar los resultados hemos observado que los más aptos para nuestro propósito son únicamente dos: Ipyleaflet y Folium.

Tanto Ipyleaflet como Folium cumplen todos los requisitos que necesitamos para el desarrollo del proyecto. Ambos son interactivos, permiten hacer mapas coropléticos y su documentación es muy completa, por lo tanto, existen múltiples ejemplos que poder usar para implementar este módulo.

Aunque las dos librerías se basen en lo mismo, usan Leaflet como base, y tienen muchas cosas en común, vamos a analizar sus diferencias.

La diferencia más significativa entre ellas es que Ipyleaflet es un widget, y está enfocado a su uso dentro de Jupyter, siendo complicado su uso fuera de los *notebooks*. Mientras que Folium es una librería, la cual se puede utilizar tanto dentro como fuera de Jupyter. Además, Folium genera finalmente un HTML interactivo que se puede exportar, para visualizarlo *offline*.

Dado que nuestro objetivo es no limitar el uso de la librería que implementamos únicamente al ámbito de Jupyter, y a pesar de que, durante un periodo de tiempo, se comenzó a implementar en Ipyleaflet. Finalmente, se concluyó en desarrollar nuestro módulo usando Folium para la representación de datos geográficos y mapas.

[18] Capítulo 2: Estado del Arte.

4. COUNTRY CODES

Los Country Codes[19] (CC) son códigos reconocidos internacionalmente que designan a cada país y a la mayoría de las áreas dependientes como una combinación de dos o tres letras; es un acrónimo, que significa un país o un estado.

- CC ISO Alfa-2: Se refiere a la combinación de dos letras.
- CC ISO Alfa-3: Se refiere a la combinación de tres letras.

El CC ISO Alfa-2 se usa, por ejemplo, para los sufijos de dos letras en dominios de nivel superior (para países) en internet. Por ejemplo: *".us"* (Estados Unidos), *".fr"* (Francia) ó *".es"* (España), entre otros.

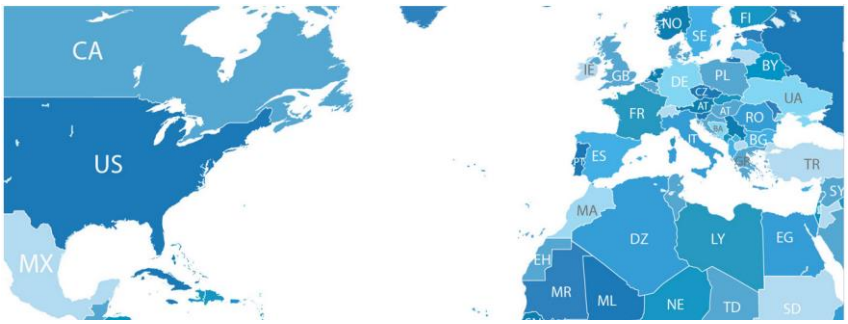

Ilustración 17. Ejemplo de mapa con Country Codes ISO Alpha-2.[20]

El CC ISO Alfa-3 se usa a menudo en eventos con participantes internacionales de diversos países, como reuniones deportivas o concursos de canciones, dónde podemos encontrar abreviaturas como: NZL (Nueva Zelanda), CUB (Cuba), PRT (Portugal), entre otras.

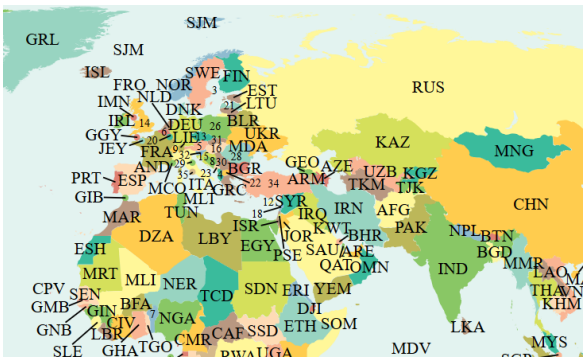

Ilustración 18. Ejemplo de mapa con Country Codes ISO Alpha-3.[21]

[19] *Country Code (CC).* Código de país.
[20] Captura de pantalla de https://www.nationsonline.org/oneworld/country_code_list.htm.
[21] Captura de pantalla de https://commons.wikimedia.org/wiki/File:ISO_3166-1_Alpha-3_map.svg

5. GEOHASHES

Un Geohash es un sistema de geolocalización que se basa en encriptar una latitud y una longitud con otro tipo de valores, estos valores están sacados de un sistema de división de cuadriculas.

Es decir, consiste en dividir todo el globo terráqueo en cuadrículas con un total de 32 celdas, estas formadas por 4 filas y 8 columnas. Cada una de las celdas se divide de nuevo en otras 32 celdas, 4 filas y 8 columnas, y así, sucesivamente. Hasta llegar al punto de interés que se desea obtener.

Cuanto se dividan las celdas, la precisión de la geolocalización es mayor.

No es un sistema muy conocido, pero en ocasiones nuestro conjunto de datos puede estar codificado en este sistema.

Este sistema de codificación consigue representar un punto del globo terráqueo mediante una cadena de texto, en lugar de un sistema de coordenadas, como es habitual, por lo tanto, el algoritmo de Geohashes usa una variante de la base 32, con una combinación de estos 32 caracteres alfanuméricos:

0123456789abcdefghjkmnpqrstuvwxyz

Estos valores una vez están representados en base 32 pueden ser convertidos a un valor binario.

Ilustración 19. Representación de códigos Geohash sobre mapa.[22]

[22] Captura de pantalla de https://www.movable-type.co.uk/scripts/geohash.html

Para lograr utilizar Geohashes como medio de representación de nuestros datos, hemos importado también la librería Geohash (Geohash2 en este caso, que es la que es compatible con Python3). Esta librería hace conversiones de longitud, latitud a Geohashes y viceversa, así es como la utilizamos. Convirtiendo el valor alfanumérico del Geohash en un conjunto (longitud, latitud) y de esta manera, lo representaremos en un mapa coroplético mediante la librería Folium.

6. OBJETIVOS

El objetivo en conjunto del módulo es la implementación de interfaces de alto nivel para la realización de visualizaciones de datos geoespaciales contenidos en Dataframes. Se busca una representación intuitiva e interactiva de un juego de datos.

El lenguaje de programación para su implementación es Python y el formato de entrada de los datos será modular y personalizable, podremos introducir datos, principalmente mediante Dataframes de la librería Pandas (por ejemplo, formato CSV[23] o JSON).

Las funciones implementadas deben aprovechar las librerías de bajo nivel ya disponible. El objetivo es potenciar dichas librerías para que se adapten a nuestros casos de uso.

El caso de uso que motiva la realización del trabajo es la falta de estas interfaces al realizar análisis de datos de publicidad online con *targeting*[24] geográfico de los usuarios. Por ello, la librería contará al menos con funciones que permitan su uso en este contexto, pero teniendo en cuenta que se puedan aplicar fácilmente para otros análisis distintos con datos de entrada en el mismo formato.

[23] CSV. Archivo delimitado por comas.
[24] Segmentación de clientes.

7. DISEÑO

En esta sección tenemos que destacar dos puntos, en primer lugar, hablaremos de unos requisitos que se deben de cumplir para que todas nuestras funciones se puedan utilizar, y, en segundo lugar, detallamos las funciones contenidas en el módulo.

7.1. Requisitos generales

El módulo Python implementado tiene que contar en el entorno del intérprete Python con las librerías Pandas y Folium ya instaladas, de no ser así no se podrá instalar este módulo, ya que estas librerías están declaradas en el archivo *requeriments.txt* que contiene las librerías indispensables para su ejecución.

Durante el proceso de instalación, se pueden instalar también las dependencias tirando del catálogo de módulos PIP [25].

7.2. Funciones

De forma más concreta, las funciones que se implementarán deberán permitir representar:

1. Puntos de interés en una representación geográfica (mapa) a partir de un Dataframe. Se representarán de forma simple y rápida puntos de interés que estén indicados en un archivo Dataframe. También se permite de forma opcional agrupar los puntos cercanos (*cluster*[26]).

2. Mapas coropléticos a partir de un Dataframe. Se representarán de forma automática mapas que contengan información estadística, que pueda ser mostrada con escalas de color, por ejemplo, un mapa que pinte densidad de población en los países.

3. Puntos de interés a partir de enlaces URL. Se representarán de forma automática puntos de interés en un mapa conociendo únicamente una URL que aporte información de dicho punto. También permite de manera opcional agrupar los puntos cercanos.

[25] *PIP*. Gestor de paquetes de Python.
[26] *Cluster.* Conjunto de objetos del mismo tipo o con características similares.

4. Puntos de interés desde coordenadas Geohash, ubicamos en una representación geográfica partiendo de un Dataframe que contiene información Geohash, traduciendo a coordenadas de latitud-longitud. Además, se muestra el radio de error de precisión para cada Geohash.

5. Áreas de densidad a partir de coordenadas latitud-longitud. De forma que se muestren círculos o iconos más grandes si hay varios puntos próximos que se podrían solapar en el mapa. Esta representación es abierta a otras opciones que permitan representar el mismo objetivo.

8. INFRAESTRUCTURAS

Para el desarrollo de este módulo se han comparado todas las posibles soluciones de producto y de hardware, para ver cuál es la más óptima y eficiente donde poder construir el entorno, descartando por el camino el montar la solución en una máquina virtual.

Hemos descartado la utilización de una máquina virtual, tras muchas pruebas previas, para poder dotar al desarrollo de portabilidad. Es decir, poder desplazar el proyecto de un equipo a otro, sin atarnos a ningún sistema operativo específico.

Ante esta decisión, se han creado varias instancias en la nube de Amazon, para aprovechar todo el potencial que nos ofrece esta tecnología.

El principal motivo para la toma de esta decisión ha sido tener en cuenta la facilidad para crear máquinas y entornos en cuestión de escasos segundos. También se ha considerado la alta disponibilidad (HA[27]) que esto ofrece, al estar siempre y en todo momento accesible desde cualquier equipo y dispositivo. De esta manera, hacemos posible ejecutar el resultado final hasta en un dispositivo móvil.

La plataforma elegida, entre las muchas opciones que existen, en la llamada “solución Cloud” hemos elegido Amazon Web Services, ya que con sus planes gratuitos nos permite tener hasta dos instancias de la maquina levantadas, y accesibles a cualquier hora del día, todos los días de la semana (24-7).

Además, entre estos planes entra la opción de crear snapshots[28], que se pueden utilizar como Back-ups de la máquina para recuperar la misma ante cualquier fallo en el desarrollo a un punto anterior.

A todas estas instancias se accede a través de una conexión de terminal SSH [29] común.

Tras elegir y montar la instancia con el Sistema Operativo, se ha instalado el motor Anaconda de Python, necesario para lanzar el Jupyter Notebook, junto con la librería Folium y Pandas de Python.

[27] HA. Por las siglas *High Ability* en inglés.
[28] *Snapshots*. Capturas o imágenes de respaldo.
[29] SSH**.** *Secure SHell* protocolo cuya función principal es el acceso remoto a un servidor por medio de un canal seguro con toda su información cifrada.

9. DESARROLLO

9.1. Pruebas funcionales

Con la infraestructura montada, hemos comenzado la lectura, el estudio de la librería Folium, ejemplos y test funcionales.

Como primer test hemos generado un mapa simple de la comunidad de Madrid:

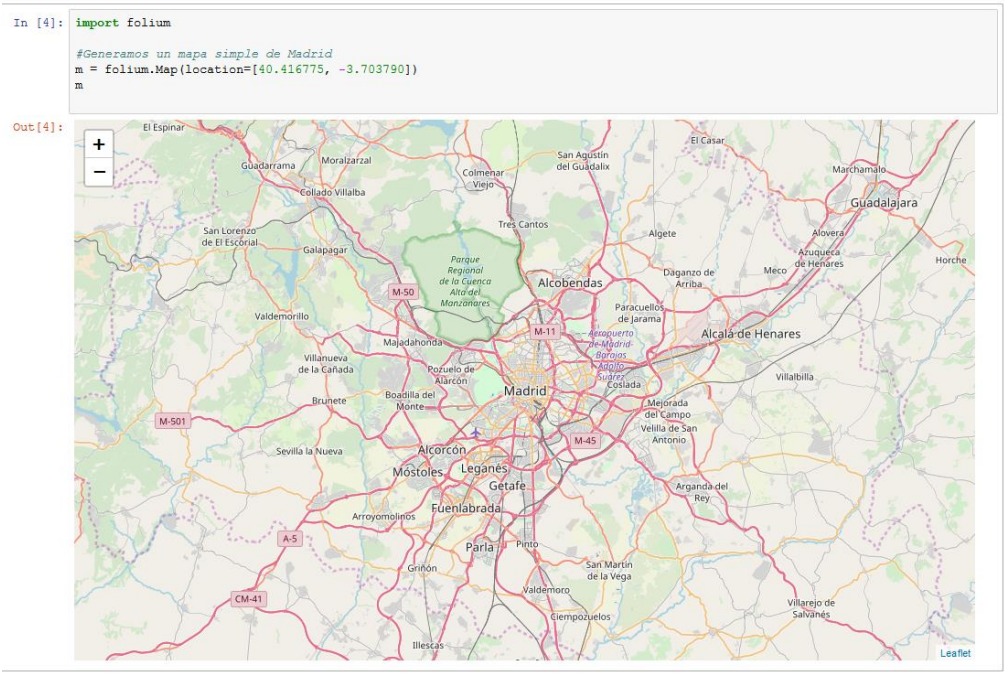

Ilustración 20. Representación del mapa de Madrid

La siguiente funcionalidad probada ha sido mostrar un punto específico en un mapa, con una “chincheta”.

En esta prueba se ha marcado la Universidad Politécnica Carlos III de Leganés, probando a su vez funcionalidades útiles como son:

- El Tooltip (lo que se muestra cuando pasas el cursor por encima).
- El Popup (lo que se muestra cuando se hace clic sobre el punto deseado).

```
In [10]: #Creamos un marcador simple
m = folium.Map(
    location=[40.331513, -3.764931],
    zoom_start=12,
    tiles='Stamen Terrain'
)

tooltip = 'Universidad Carlos III - Leganes'
folium.Marker([40.331513, -3.764931], popup='Universidad Carlos III - Leganes -- España', tooltip=tooltip).add_to(m)
m
```

Out[10]:

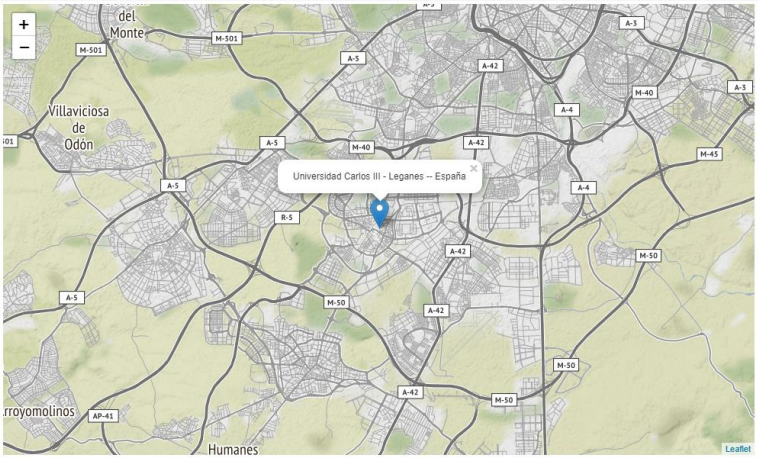

Ilustración 21. Representación de puntos interés.

La siguiente prueba realizada es la selección de un área de acción sobre el mapa, esto puede ser útil si se desean comparar dos áreas diferentes mediante un análisis de datos.

En este caso, seleccionamos el área de la Universidad Carlos III en dos de sus principales campus[30].

```
In [23]: #Creamos un mapa con un estilo (tiles)
m = folium.Map(
    location=[40.311358,-3.726505],
    zoom_start=13
)
#Creamos una sección con circulo
folium.Circle(
    radius=100,
    location=[40.332544,-3.765349],
    popup='Leganes - Universidad Carlos III',
    color='crimson',
    fill=False,
).add_to(m)

#Creamos otro tipo de circulo - marcado
folium.CircleMarker(
    location=[40.317101,-3.727541],
    radius=15,
    popup='Getafe - Universidad Carlos III',
    color='#3186cc',
    fill=True,
    fill_color='#3186cc'
).add_to(m)

#Mostramos el entorno
m
```

Out[23]:

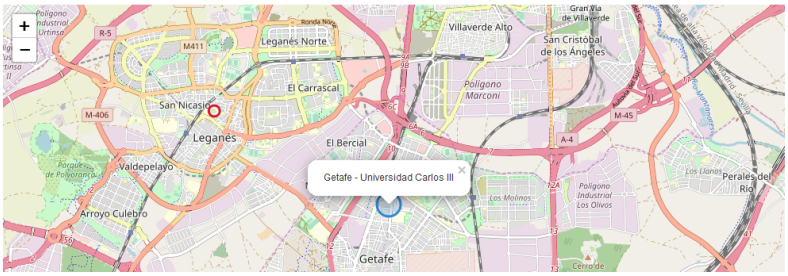

Ilustración 22. Representación de áreas de acción.

[30] Universidad Carlos III - Campus C/ Madrid, 126. 28903 Getafe, Madrid
Universidad Carlos III - Campus Avda. Universidad, 30. 28911 Leganés, Madrid

Después de realizar estas pruebas básicas, como poner marcadores o seleccionar áreas concretas, empezamos a trabajar con la entrada de datos procedentes de archivos reales en formato Dataframe.

En este ejemplo, lo haremos con España, utilizando como puntos de interés los aeródromos y aeropuertos de todo el país.

El punto de entrada de información de este mapa es un Dataframe (archivo .csv) con los aeropuertos que nos interesaría mostrar, añadiendo un Popup por cada uno de los puntos de interés.

```
In [127]: import folium

#Importamos el Dataset
f = open('spain_airports.csv')
reader = csv.reader(f)

#Creamos un mapa con muchos marcadores
m = folium.Map(
    location=[40.331513, -3.764931],
    zoom_start=12,
    tiles='Stamen Terrain'
)

#Recorremos todo el Dataset añadiendo una marca por cada aeropuerto
for row in reader:
    lat = float(row[4])
    lon = float(row[5])
    folium.Marker(
        location=[lat, lon],
        popup=folium.Popup(str(row[3]), parse_html=True, max_width=100)
    ).add_to(m)

m
```

Out[127]:

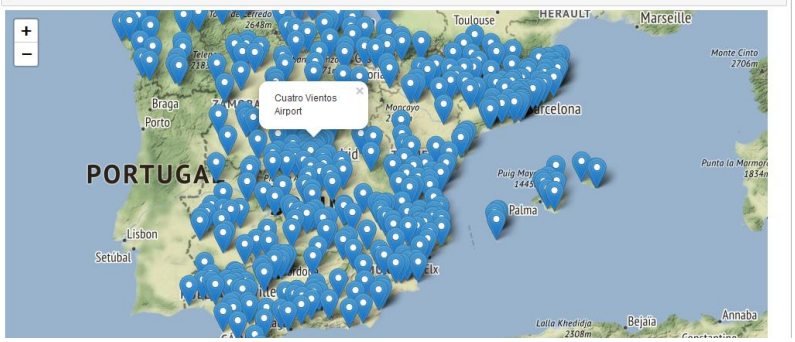

Ilustración 23. Representación de los aeródromos de España.

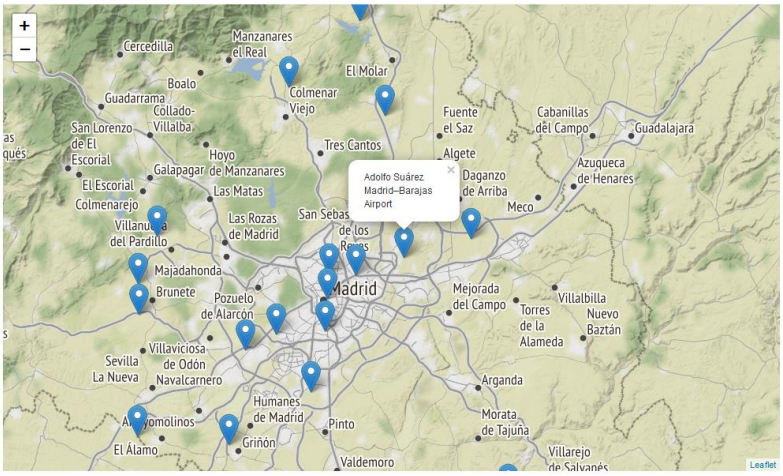

Ilustración 24.Representación de Aeródromos y aeropuertos de Madrid.

Continuando con el desarrollo, nos hemos basado en la herramienta Overpass turbo[31] para construir un geoJSON, que es un archivo que delimita todas las zonas de actuación del mapa interactivo que se pretende crear.

La representación creada es un mapa cloroplético con identificadores de colores según valores estadísticos. Que se interpreta en el Dataframe como un dato, que posee un valor, el cual corresponde en una escala de valores, y se representa como un color en una escala cromática seleccionada.

Como punto de entrada tendríamos un archivo Dataframe (.csv), que contiene cifras de usuario únicas por país en el año 2017.

Para jugar un poco con las posibilidades de esta funcionalidad hemos acotado el archivo csv, para seleccionar solamente aquellos países que nos interesaban.

```
In [34]: import pandas as pd
         import folium
         #Seleccionamos el archivo geojson
         countries = 'world-countries.json'

         #Seleccionamos el archivo datasheet
         users = pd.read_csv('internet_users.csv')
         users

         #Creamos un mapa
         mapa = folium.Map(location=[40.420423, -3.696685], zoom_start=6)

         #Creamos las propiedades coropléticas
         mapa.choropleth(
             geo_data=countries,
             name='choropleth',
             data=users,
             columns=['Code', 'Users'],
             key_on='feature.id',
             fill_color='YlGn',
             fill_opacity=float(0.7),
             line_opacity=float(0.2),
             legend_name='Usuarios de Internet 2017'
         )
         folium.LayerControl().add_to(mapa)
         mapa
```

Out[34]:

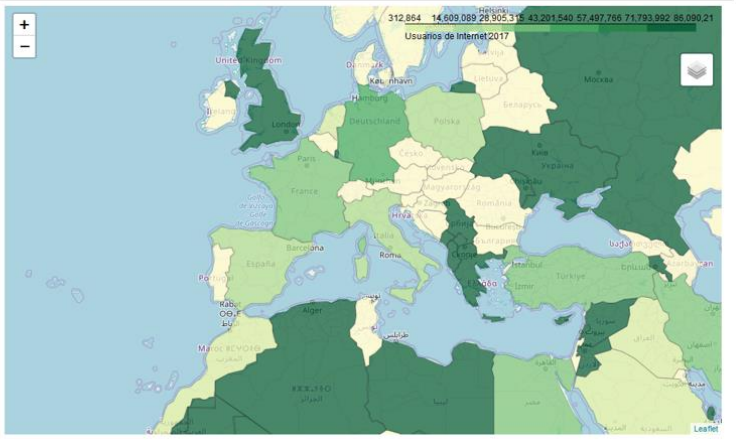

Ilustración 25. Representación de los usuarios de internet 2017 por países.

[31] Overpass turbo. https://overpass-turbo.eu/

Ilustración 26. Representación de usuarios únicos de internet 2017 en España.

9.2. Construcción de recursos

9.2.1. Overpass Turbo

La API Overpass es una gran herramienta para el mapeo, ya que es muy potente para filtrar datos OSM[32]. Mediante el modo de Overpass turbo, existe una manera fácil de ejecutar rápidamente cualquier consulta de Overpass e inspeccionar los resultados de un modo muy simple para el usuario.

Overpass turbo principalmente es una herramienta útil para el público genera, para filtrar ciertas cosas que se están buscando.

Pero existen varias ideas para las que usar Overpass turbo, de forma que sea una herramienta bastante útil para el mapeo:

- Buscar POI[33] especiales que no está dibujados en el mapa.
- Inspeccionar los POI que se distribuyen uniformemente en grandes áreas.
- Mostrar características espaciales grandes (parques, ríos, límites, autopistas completas, estaciones de servicios, zonas de turismo…) y cargarlas directamente en un editor.
- Utilizar solo una porción filtrada de los datos OSM.

Para desarrolladores, Overpass Turbo ofrece algunas opciones:

- Permite probar y desarrollar consultas de API Overpass más o menos complicadas.
- Convierte datos OSM al formato de datos geoJSON.

[32] OSM. Una herramienta de línea de comandos que se utiliza para filtrar archivos de datos de *Open Street Map* para etiquetas específicas.

[33] POI. Puntos de interés (*Point of interest*).

- Crea maquetas de mapas en los que es posible hacer clic o estáticos, en los que se pueden resaltar las características seleccionadas de OSM.

Una curiosidad que tiene Overpass Turbo es que es muy conocido para jugadores de Pokémon GO[34], que utilizan esta herramienta para elaborar rutas para posibles encuentros, para buscar nidos y también, para encontrar EX Raids[35].

Para consultar en Overpass turbo, se coloca la consulta de Overpass API en el editor, se presiona el botón Ejecutar y se espera a que aparezca el juego de datos OSM que se muestran, en ocasiones esta espera es muy larga, se puede modificar el tiempo de espera en milisegundos en el código que aparece a la izquierda. Tiene un asistente de consultas, el cual, está diseñado para transformar términos de búsquedas simples y legibles para el usuario, en consultas funcionales de Overpass. Alternativamente te deja escribir una etiqueta, para realizar la consulta.

Overpass turbo muestra la mayor cantidad de datos posibles y al hacer clic en un objeto una ventana emergente muestra toda la información del punto seleccionado. También distintos tipos de identificación, etiquetas, coordenadas y metadatos, si están disponibles para nuestra consulta.

Es una herramienta bastante completa, que nosotros usaremos para la búsqueda de POI que será devuelta en formato de datos geoJSON.

Para utilizarla, hace falta un navegador web reciente y actualizado, hemos probado con Opera, Chrome y Firefox, con todos ellos funciona correctamente.

9.3. Entorno de desarrollo integrado (IDE)

Como entorno de desarrollo principal se ha decidido utilizar PyCharm por ser uno de los entornos de desarrollo más conocidos y utilizados al desarrollar con Python.

Trabajar con un entorno de desarrollo integrado (IDE) tiene muchas ventajas, la principal es que se trata de un editor de código que puede servirnos para depurar y facilitarnos las diferentes tareas necesarias en el desarrollo de cualquier tipo de aplicación.

[34] Pokémon GO. https://www.pokemongo.com/es-es/

[35] *EX-Raids*. Incursiones Exclusivas de Pokémon GO. https://pokemongohub.net/gym-raid-update/comprehensive-guide-trigger-ex-raids/

Este entorno de desarrollo integrado nos permite también validar el código en tiempo real ya que nos indica errores y posibles mejoras del código, mostrando ayuda a documentación oficial en muchos de los casos.

9.3.1. PyCharm vs Python IDE

Si hablamos de edición de código, el IDE integrado que ofrece Python al instalarse es bastante bueno y simple para edición de código, sin embargo, no nos ofrece todas las ventajas que nos ofrece de serie PyCharm, teniendo que acudir a otras herramientas para realizar distintas tareas como puede ser crear módulos o paquetes.

9.3.2. PyCharm vs Visual Studio Code

Un editor de código que está cogiendo mucha fuerza últimamente es la versión gratuita y Open Source del famoso editor de Microsoft, Visual Studio Code. Esta versión es multiplataforma y aporta muchas más funcionalidades que el IDE por defecto de Python, entre las que destacamos la consola terminal integrada y la funcionalidad "*intellisense*"[36] que nos ayuda a crear un código limpio, eficiente y de calidad. Sin embargo, este entorno también se nos queda corto ya que sirve para editar archivos .py únicos y no paquetes o módulos que es lo que necesitamos.

Por todo esto, nos decantamos por utilizar PyCharm, tras elegirlo, se ha realizado la instalación y configuración del entorno.

Todo esto nos ha facilitado el trabajo para depurar el código de la aplicación y encontrar errores, corriendo secciones del código y probando sentencias con la consola integrada del entorno. Gracias al entorno de desarrollo se ha podido llevar un control de versionado en un repositorio en la nube (GitHub), para recuperar secciones de código antiguas sin problemas y realizar un control de cambios eficiente acorde a las metodologías de programación más recomendadas. Esta decisión del control de versiones va ligada a la decisión de usar una infraestructura en la nube Amazon Web Services (AWS) y su control de capturas, versiones y back-up.

[36] *Intellisense.* Ayuda de autocompletado de nombres de funciones, atributos, paths, …

9.4. Creación de proyecto y configuración

Después de configurar y abrir PyCharm iniciamos un nuevo proyecto que vamos a llamar **data2maps**. La creación de este proyecto automáticamente nos generará un **venv**[37] listo para instalar todos los paquetes que necesitaremos para nuestro desarrollo (en este caso, Pandas y Folium).

Tras instalar los paquetes, estará disponible todo lo necesario en nuestro desarrollo para exportar la aplicación con sus dependencias.

Una vez desarrolladas todas las funciones utilizamos el propio PyCharm y la librería PyBuilder para compilar el paquete de funciones como una librería independiente e instalable para Python.

9.5. Generar y publicar la librería

Ya que nuestra intención es hacer que la librería implementada sea pública y accesible, para el resto de los desarrolladores de Python, debemos empaquetarla en un módulo aparte y subirla a un repositorio público. El repositorio más conocido actualmente para este fin es el PyPI[38], donde una vez publicado, será accesible desde el comando ***pip install***[39].

Para poder comprimir y empaquetar la librería creada en un módulo que se pueda publicar, necesitamos una herramienta que se encargue de ello, en este caso hemos optado por utilizar la librería PyBuilder.

PyBuilder se encarga de hacer un empaquetado de manera automática, después de ejecutar nuestro conjunto de pruebas, de observar que compila y de comprobar que todo está correcto.

Puesto que estamos utilizando PyCharm, el proceso general de la instalación es mucho más sencillo que si se hiciera directamente por consola, ya que esta librería se encarga de gestionar el ***venv*** con la versión de Python elegida, y con todas las dependencias en un entorno aislado.

[37] *Venv.* Un entorno virtual de Python independiente del mismo e instalado en el sistema.
[38] PyPI. *Python Package Index.* Repositorio de paquetes de Python.
[39] *Pip install.* Comando de Python para instalar dependencias.

Primero descargaremos la dependencia de PyBuilder, dentro de PyCharm. En las configuraciones se encuentra la opción “Project Interpreter” donde seleccionaremos nuestra versión de Python a utilizar que automáticamente nos instala los módulos de ***pip***[40], ***setuptools***[41] y ***wheel***[42].

Mediante el botón de instalar podremos buscar PyCharm e instalarlo.

A continuación, crearemos el fichero ***build.py*** en la raíz del proyecto, y generar la estructura de carpetas que PyBuilder necesita. Por convención, la carpeta con nuestro código fuente se debe de encontrar en `'src/main/python'`. Una vez tenemos nuestro código en dicha carpeta tendremos que indicar a través de ***build.py*** la configuración básica para compilar la librería. Cuando lanzamos el comando ***pyb***[43], que nos creará la carpeta ***dist*** [44]con la primera versión del paquete listo para ser publicado.

Al realizar la compilación, el propio PyBuilder lanza una comprobación y depuración de todas las funciones, realizando un análisis heurístico que comprueba que todas las funciones están referenciadas, que todos los parámetros pueden ser utilizados y que todas las instrucciones son correctas.

El paquete resultante del proyecto puede ser publicado en un repositorio público, privado o se puede utilizar dentro de otro proyecto Python de manera local.

Para probar que nuestro paquete funciona correctamente debemos llevarnos los ficheros de ***dist*** a otro proyecto Python y ejecutar el archivo llamado **setup.py** (archivo de instalación), y de este modo, se pueden llamar a las funciones de nuestro módulo en otro proyecto distinto.

Además, se han añadido dependencias para generar código limpio mediante el uso de linters[45], de un mecanismo de generación automática de documentación, y de una comprobación de la cobertura de test, cuidando que supere un umbral del 70%.

Por último, para publicar la librería necesitamos estar registrados en PyPI, en mi caso el usuario es **IsabelaSJ**. Con la ayuda de la librería ***twine***[46] podemos publicar el módulo compilado usando nuestras credenciales.

---

[40] *Pip.* Instalador de paquetes de Python.
[41] *Setuptools.* Biblioteca de procesos de desarrollo de paquetes de Python.
[42] *Wheel.* Formato de empaquetamiento de Python.
[43] *Pyb*.Comando básico de PyBuilder.
[44] *Dist.* Directorio de destino al construir un paquete de Python.
[45] *Linters.* Herramienta que analiza el código fuente para marcar errores de programación.
[46] *Twine.* Comando para publicar paquetes de Python en PyPI.

9.6. Producto final

El producto final consta de un módulo publicado en el repositorio público de PyPI, junto con una documentación que detalla para cada una de las funciones, los atributos que se esperan y su forma de ser utilizados, incluyendo ejemplos con código explicando cómo usar el módulo **data2maps**.

A continuación, se detalla el alcance del módulo y las funciones implementadas.

9.6.1. Puntos de interés

La primera función que encontramos en el módulo es **poi.** Esta función recibe por parámetro un Dataframe o un fichero delimitado por comas (csv) cuyas tuplas deben contener tres parámetros clave:

- Latitud
- Longitud
- Nombre o descripción

Los nombres de las columnas que contienen dichos atributos se pasan como segundo parámetro en un array llamado "*columns*".

El módulo recorre todos los registros del dataframe y recoge cada uno de los puntos de interés que encuentre en él, comprobando en todos los casos que contiene los atributos de latitud, longitud y nombre. Y después, generará un mapa en dos dimensiones en el que posicionará todos los POI encontrados, mostrando en su ubicación un puntero con un tooltip con el nombre o descripción.

A partir de la ubicación media de todos los puntos de interés encontrados, centramos el mapa automáticamente.

Esta función acepta como parámetros opcionales de tipo *Boolean*[47]:

- *Clustered.* Si se encuentra activado, a la hora de visualizar el mapa se agruparán los POI cercanos, pudiéndolos distinguir de manera separada al hacer zoom sobre ellos.
- Write_html. Si se encuentra activado, guardará en formato HTML dinámico e interactivo el mapa, el cual se podrá visualizar una vez se termine de escribir.

[47] *Boolean.* Tipo de parámetro de programación que puede tomar valores True o False, dependiendo si está activado o desactivado.

Ilustración 27. Representación de puntos de interés “not clustered”.

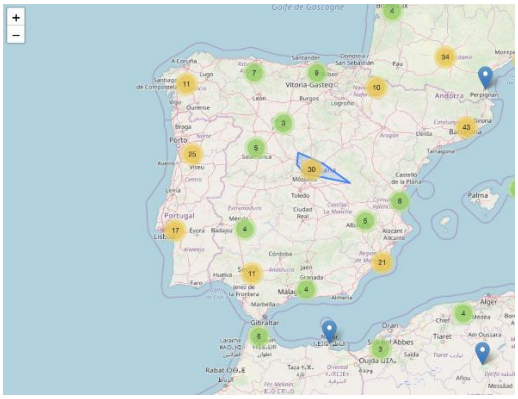

Ilustración 28. Representación de puntos de interés “clustered”.

9.6.2. Mapa cloroplético

Un mapa cloroplético consiste en la representación visual en un conjunto de datos geográficos mostrados en una escala de color según sus valores. Un mapa cloroplético tiene un uso muy extendido en la representación de datos demográficos, en mapas de meteorología, en mapas topográficos y en la representación de datos estadísticos geolocalizados, en este último es en el que enfocaremos nuestros ejemplos, aunque nuestra librería será útil también para los otros casos.

La siguiente función implementada en el módulo es **choropleth**. Esta función es capaz de representar un mapa cloroplético obteniendo cómo parámetros de entrada un Dataframe y las tonalidades con las que se desea representar en el mapa, en este caso se puede elegir entre tres tipos distintos de tonalidad: verde, azul o rojo.

La función recorre todas las tuplas del Dataframe recibido y añade al mapa los puntos geográficos de interés, mostrando según su escala de valores, su tonalidad correspondiente, tonos más claros para valores bajos y tonos más oscuros para valores altos.

A partir de los datos del geoJSON se calculan los límites de representación del mapa, donde se centrará el mapa automáticamente.

Esta función acepta como parámetro opcional de tipo *Boolea*n:

- Write_html. Si se encuentra activado, guardará en formato HTML dinámico e interactivo el mapa, el cual se podrá visualizar una vez se termine de escribir.

De este modo, finalmente se consigue una representación de un mapa en el que todos los datos reciben un color, dependiendo de las zonas de influencia de los datos.

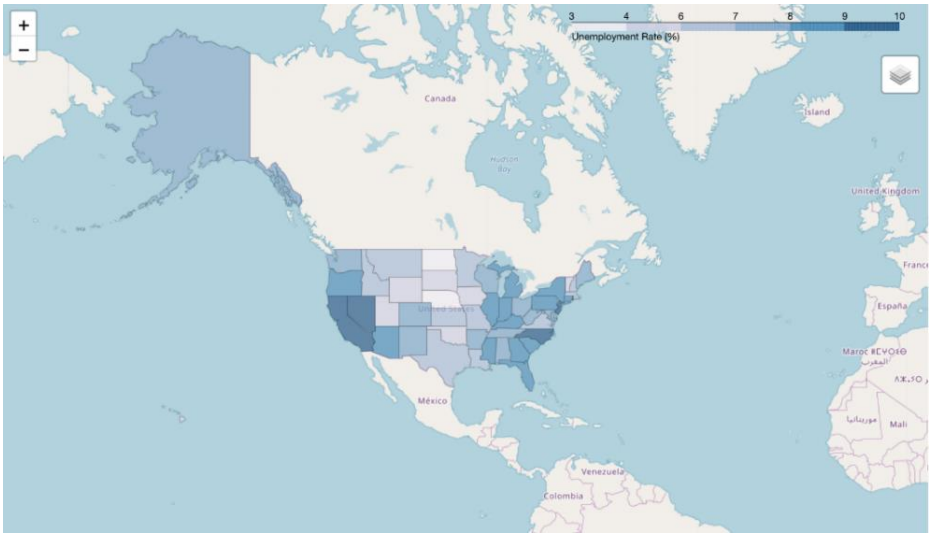

Ilustración 29. Representación geográfica generada mediante la función choropleth.

9.6.3. Parseador url de Google Maps

El uso de Google Maps está muy extendido en todos los ámbitos, una gran parte de la comunidad que trabaja con datos geográficos, en ocasiones, almacena la url de Google en bruto, en lugar del punto con coordenadas GPS[48]. Por eso consideramos que es muy importante poder interpretar este formato también. Además, puede ser muy útil acceder a los metadatos que Google nos provee a través de su API[49]; por ejemplo, nombre del

[48] *GPS. Global Positioning System*. El Sistema de Posicionamiento Global.
[49] *API*. *Application Programming Interface.* Interfaz de programación de aplicaciones.

lugar marcado, tipo de localización que es, o incluso la puntuación del lugar votada por los usuarios.

La función implementada para este caso de uso se llama **gmaps_url**. Es capaz de interpretar la url de Google y extraer sus coordenadas, además de obtener también el nombre del punto de interés.

Por ejemplo, a partir de esta url:

https://www.google.com/maps/place/Universidad+Carlos+III+de+Madrid:+Campus+de+Getafe/@40.3170572,-3.7296565,17z/data=!3m1!4b1!4m5!3m4!1s0xd4220b90a840de7:0x8bc24e4eab60c842!8m2!3d40.3170572!4d-3.7274678

Se parseará parar recibir la información del punto de interés que viene contenida en la misma, en este caso de ejemplo "*Universidad Carlos III de Madrid: Campus de Getafe*" y recibir también las coordenadas en formato latitud y longitud, que viene tras el @ como: *40.3170572, -3.7296565*.

Esta función acepta como parámetros opcionales de tipo *Boolea*n:

- *Clustered.* Si se encuentra activado, a la hora de visualizar el mapa se agruparán los puntos de interés obtenidos a partir de las url de Google, pudiéndolos distinguir de manera separada al hacer zoom sobre ellos.
- Write_html. Si se encuentra activado, guardará en formato HTML dinámico e interactivo el mapa, el cual se podrá visualizar una vez se termine de escribir.

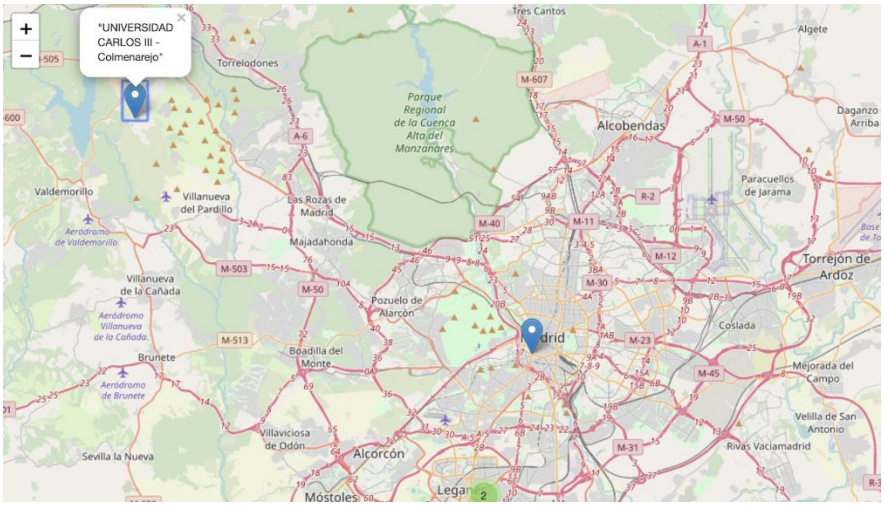

Ilustración 30. Representación geográfica generada mediante la función gmaps_url.

9.6.4. Geohash

Un Geohash es un código alfanumérico único que representa una coordenada en latitud y longitud con un formato más liviano y que alivia el peso de los archivos Dataframe, importante cuando tratamos con Dataframes con inmensa información, pudiendo reducir el tamaño hasta un 65%.

Dependiendo de la longitud del Geohash aumentará o disminuirá la precisión, por ejemplo, con 11 caracteres obtendremos la precisión de ±7,4 centímetros, mientras que con un sólo caracter tendremos una precisión de ±2.500 km.

La última función implementada, es la llamada **geohash**, que consiste en la interpretación de la encodificación Geohash.

A partir de un Dataframe recibido por parámetro iremos descodificando los distintos Geohashes en latitud y longitud, y posteriormente los añadiremos al mapa al igual que en el resto de las funciones.

En esta función en lugar de representar un punto, o un conjunto de puntos, como se hace en las funciones anteriores, representaremos un área de forma circular, cuyo radio representa el error de precisión en función a la longitud de la cadena del Geohash.

Esta función acepta como parámetro opcional de tipo *Boolean*:

- Write_html. Si se encuentra activado, guardará en formato HTML dinámico e interactivo el mapa, el cual se podrá visualizar una vez se termine de escribir.

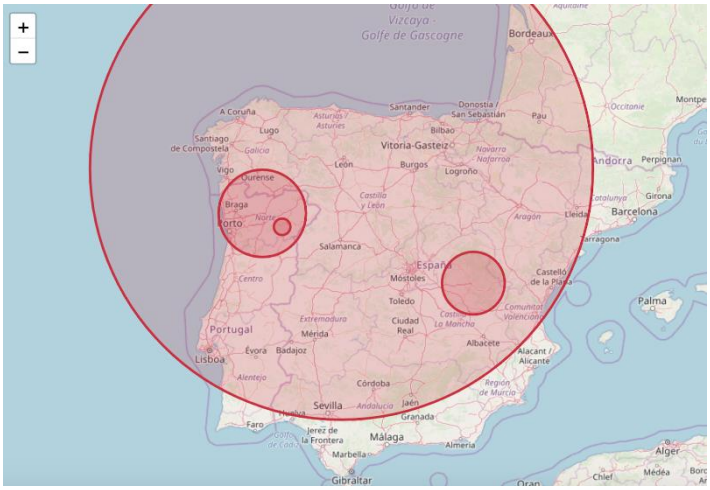

Ilustración 31. Representación de 4 puntos Geohash mediante la función geohash.

10. ASPECTOS ECONÓMICOS

En este capítulo se van a detallar los aspectos económicos del proyecto y el análisis de los costes de desarrollo, que serán necesarios para terminar el módulo en los tiempos establecidos.

Los costes están desglosados en recursos de personal, que implica que todos los perfiles que constan en él están involucrados en el proyecto, coste de hardware, software y licencias, y, por último, se establecen los costes indirectos.

Se ha tenido en cuenta un riesgo del 30%, ya que basándonos en el ratio de éxitos de proyectos similares, se ha estimado un porcentaje de incapacidad de finalización del proyecto, debido a diversos motivos (desde que el cliente se declare insolvente de pago, hasta que ocurra algún tipo de percance con los desarrolladores). También dentro de este porcentaje quedan incluidos los costes debidos a la dilatación del proyecto por encima del tiempo estimado.

Las ganancias fijadas en un 15%, se han calculado estimando el porcentaje de beneficio que obtenemos tras la finalización exitosa del proyecto. Esta finalización debe de cumplir los plazos fijados.

10.1. Costes de hardware, de software y de personal

Coste	Precio	Tiempo	Total	Descripción
Internet	48 €/mes	7 meses	336 €	ADSL 100 MB
Lugar de trabajo	283 €/mes	8 meses	2.264 €	Espacio de trabajo *co-working*[50]
Servidor Amazon	0,0416 €/hora	800 horas	33,28	T3.medium + 4GB
Hardware				
PC	1.300 €/3años	8 meses	288,9 €	ASUS ROG core i7 32GB RAM
Software				
Licencia Microsoft	75,95 €/año	8 meses	50,6 €	Microsoft Office 365
Licencia PyCharm	200 €/año	6 meses	100 €	PyCharm Professional Developers
Personal				
Desarrollador	24 €/hora	800 horas	21.000 €	Salario

Tabla 1. Tabla que refleja los costes de hardware, software y personal durante el desarrollo del proyecto.

TOTAL	**24.072,78 €**
RIESGO (30%)	7.221,83 €
GANANCIAS (15%)	3.610,92 €
TOTAL SIN IMPUESTOS	34.905,53 €
TOTAL CON IMPUESTOS (21%)	**42.235,69 €**

Tabla 2. Tabla con el total de los costes, riesgos, ganancias y total con impuestos del proyecto.

[50] *Co-Working.* Una forma de trabajo que permite a profesionales independientes, emprendedores, y pymes de diferentes sectores, compartir un mismo espacio de trabajo y equipamientos, para ahorrar gastos y fomentar relaciones sociales estables entre ellos.

11. LÍNEAS FUTURAS

En este capítulo se pretenden describir futuras opciones que nos han surgido tras ir aprendiendo a utilizar librerías de Python, como Folium o Pandas y tras entender lo que es un Dataframe, y dónde obtenerlos (https://overpass-turbo.eu/[51]). Después conocer cómo trabajar con ellos, hemos visto que por falta de tiempo y nivel del conocimiento del desarrollo no hemos podido implementar todas las funciones que nos hubiera gustado.

Durante el desarrollo de este proyecto se nos han ido abriendo una gran cantidad de posibilidades y muchas ideas donde aplicar todo lo aprendido, para poder potenciar nuestro módulo.

11.1. Algoritmos de reconocimiento de imágenes con Google Maps.

Una de las opciones más interesantes que hemos observado, y que por falta tiempo en el desarrollo no hemos podido implementar, es la idea de utilizar algoritmos de reconocimiento de imágenes para crear una función, la cual recibiendo como entrada una imagen (una fotografía tomada con un dron) pueda ser capaz de determinar y representar el punto en el que nos encontramos en un mapa.

11.1.1. Ejemplo de reconocimiento de imágenes con Google Maps

Si con un dron hiciéramos una fotografía del patio interior de la Universidad Carlos III y pasáramos la imagen capturada por nuestro algoritmo, se reconocería la imagen, contrastando los colores con la base de datos de conocimiento de Google Maps, y toda su información de relieve. De esta manera, se podría determinar que mediante una concordancia de más del 98%, nos pueda decir exactamente que punto estamos fotografiando y a partir de ahí comenzaría a explorar el mapa desde ese punto, con este procedimiento podríamos incluso trazar un recorrido desde la Universidad hasta nuestra casa.

11.2. Ampliación de la función de puntos de interés mediante URL

Otra línea que se puede seguir en el futuro es la mejora de algo que ya existe en nuestro módulo, concretamente mejoraríamos la función que utiliza las URL para añadir a los puntos de interés (POI) información contrastada del sitio a representar.

[51] *Véase el apartado 8.2.1*

Nos hubiera gustado añadir a los POI comerciales la información de negocio de este punto.

11.2.1. Ejemplo de ampliación de la función de POI mediante URL

Si el punto de interés fuera un restaurante, añadiríamos información del tipo de comida que ofrece dicho restaurante. Somos conscientes de que esto, ya está implementado en otras aplicaciones del mercado, pero la funcionalidad que queremos ampliar es poder guardar nuestro propio mapa de POI y retroalimentar el mismo para tener una base de datos de conocimiento alimentada.

12. CONCLUSIONES

La evidencia que se ha mostrado anteriormente demuestra que mejorar siempre es una tarea complicada que depende de nuevas herramientas o técnicas. Precisamente mediante la implementación de este módulo de Python se ha pretendido demostrar que, frente a las múltiples bibliotecas de Python que pueden sernos útiles para la representación geográfica de datos, se puede simplificar de manera concreta, creando una herramienta nueva, más precisa y acertada.

Tras observar la carencia que existe en la representación visual de datos, con este módulo se ha pretendido alcanzar el mayor potencial que se puede adquirir, para que corresponda su funcionamiento a una buena visualización, siendo atractiva y fácil al uso. Incluso sin necesidad de conocimientos de programación, simplemente instalándolo y pasándole los datos que se deseen analizar por parámetro.

Ya que existen infinidad de librerías para realizar estas visualizaciones, y todas muy complicadas de utilizar, se pretende agrupar sus funcionalidades en una única librería, que muestre a científicos de datos e investigadores una buena visualización de los resultados. Obteniendo resultados inmediatos sin grandes esfuerzos.

BIBLIOGRAFÍA

146 examples for datawrapper choropleth maps. Disponible en: https://academy.datawrapper.de/article/149-examples-of-datawrapper-choropleth-maps.

A comprehensive guide on how to trigger EX-Raids. Disponible en: https://pokemongohub.net/gym-raid-update/comprehensive-guide-trigger-ex-raids/.

About dCode and its Author - Contact Information. Disponible en: https://www.dcode.fr/about.

AIZENMAN, H., GROSSBERG, M., JONES, D., BARNES, N., SMERDON, J., ANCHUKAITIS, K. y GEAY, J.E., 2012. Web Based Visualization Tool for Climate Data Using Python. 92nd AMS Annual Meeting, Second Symposium on Advances in Modeling and Analysis Using Python. S.l.: s.n.

ÁVILA, O. y VALENTÍN, E., 2015.

BASSI, S., 2007. A primer on python for life science researchers., vol. 3, no. 11, pp. e199.

Bokeh Development Team (2018). Bokeh: Python library for interactive visualization. Disponible en : http://www.bokeh.pydata.org.

Bokeh Plotting Backend for Pandas and GeoPandas. Disponible en: https://pythonawesome.com/bokeh-plotting-backend-for-pandas-and-geopandas/.

Bokeh Pydata. Disponible en: https://bokeh.pydata.org/en/latest/docs/gallery.html.

BRAVO, J.D., 2000. Breve introducción a la cartografía ya los sistemas de información geográfica (SIG). S.l.: Ciemat.

CÁRDENAS VELÁSQUEZ, K., 2017.

CARRIEL ESPINOZA, L.A. y DELGADO BURGOS, L.F., 2017.

CASTRO, N., 2017. Introducción a data science con Python.

CRICHTON, D.J., MATTMANN, C.A., CINQUINI, L., BRAVERMAN, A., WALISER, D., GUNSON, M., HART, A.F., GOODALE, C.E., LEAN, P. y KIM, J., 2012. Sharing satellite

observations with the climate-modeling community: software and architecture. , vol. 29, no. 5, pp. 73-81.

CRICHTON, D.J., MATTMANN, C.A., CINQUINI, L., BRAVERMAN, A., WALISER, D., GUNSON, M., HART, A.F., GOODALE, C.E., LEAN, P. y KIM, J. Sharing Satellite Observations with the Climate-Modeling Community.

CUTTONE, A., LEHMANN, S. y LARSEN, J.E., 2016. Geoplotlib: A Python Toolbox for Visualizing Geographical Data.

Darwin, Ian F. 1991. *Checking C Programs with Lint: C Programming Utility* (Revised ed.). United States: O'Reilly Media. *ISBN 978-0937175309*

Data Plotly para generar gráficos interactivos en QGIS. 2018. Disponible en: http://www.geomapik.com/analisis-gis/data-plotly-plugin-generar-graficos-interactivos-qgis/.

Data Science Blog by Domino. Disponible en: http://blog.dominodatalab.com/.

Determining Which Gyms will Get you Ex-Raid Passes for Mewtwo. Disponible en: https://articles.pokebattler.com/2018/02/26/determining-which-gyms-will-get-you-ex-raid-passes-for-mewtwo-part-2/.

Díaz, S., Torres, J. M., Aguilar, R. M., & Parra, E. 2018. Cuadro de mando para un destino turístico inteligente. Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de septiembre de 2018.

DOMÍNGUEZ PARRÓN, B., 2016. Gestión del acceso remoto a kits de laboratorio de prácticas CISCO.

DOYLE, S., DODGE, M. y SMITH, A., 1998. The potential of web-based mapping and virtual reality technologies for modelling urban environments., vol. 22, no. 2, pp. 137-155.

DRABAS, T., 2016. Practical Data Analysis Cookbook. S.l.: Packt Publishing Ltd.

ECKLE, M. y DE ALBUQUERQUE, J.P., 2015. Quality Assessment of Remote Mapping in OpenStreetMap for Disaster Management Purposes. ISCRAM. S.l.: s.n.

Eliminar filas o columnas con valores nulos en Python. Disponible en: https://www.analyticslane.com/2018/06/29/eliminar-filas-o-columnas-con-valores-nulos-en-python/.

ESTUPIÑAN DUARTE, P.E. Apoyo y Soporte Técnico en la Implementación de un Sistema de Información Geográfico Mediante Software Libre, para la Secretaría de Urbanismo del Municipio de Facatativá-Cundinamarca.

FARRELL, D. y GORDON, S.V., 2015. Epitopemap: a web application for integrated whole proteome epitope prediction., vol. 16, no. 1, pp. 221.

FONTE, C.C., MINGHINI, M., ANTONIOU, V., SEE, L., PATRIARCA, J., BROVELLI, M.A. y MILCINSKI, G., 2016. An automated methodology for converting OSM data into a land use/cover map. Proceedings of the 6 th International Conference on Cartography & GIS. S.l. : s.n., pp. 462-473.

FONTS, O. Generación automática de visualizadores web 3D.

FRONTERA, B., 2016. Leaflet. TimeDimension: ¡esto se anima!

GARCÍA, A.M. y CASTAÑO DÁVILA, A.C., 2013. SIG de deslizamientos para el departamento de Caldas.

GUTIÉRREZ DUQUE, D.M., 2016. Enlaza plataforma Sabana.

HAAGSMAN, E., 2017. How to Publish Your Package on PyPI. Disponible en: https://blog.jetbrains.com/pycharm/2017/05/how-to-publish-your-package-on-pypi/.

HAFFNER, B., 2018. Folium MarkerClusters and FastMarkerClusters. Disponible en: https://medium.com/@bobhaffner/folium-markerclusters-and-fastmarkerclusters-1e03b01cb7b1.

HOLTZ, Y., 2017. #292 Choropleth map with Folium. Disponible en: https://python-graph-gallery.com/292-choropleth-map-with-folium/.

JOHNSON, S.C., 1977. *Lint, a C program checker*. Murray Hill, New Jersey: Bell Laboratories. Computing science technical report, 65.

LEFEBVRE, A.E., 2015. No title.

Managing Your Python Project with Git and PyBuilder. Disponible en: https://dev.to/awwsmm/managing-your-python-project-with-git-and-pybuilder-21if.

MAP, 2017. Disponible en: https://python-graph-gallery.com/map/.

Matplotlib Tutorial 28 - plotting coordinates on a map with Basemap. Disponible en: https://www.youtube.com/watch?v=8v3how07th4.

MÉNDEZ NOBOA, D.F., 2012.

MOONEY, P. y MINGHINI, M., 2017. A review of OpenStreetMap data.

NATIONSONLINE.ORG, klaus kästle. Country Codes List - Nations Online Project. Disponible en: https://www.nationsonline.org/oneworld/country_code_list.htm.

NOEL, S., AULT, A., WANG, Y., LAYTON, A., BALMOS, A., KROGMEIER, J.V. y BUCKMASTER, D., 2018. Real-time on-farm yield trials powered by open-source : connecting ISOBlue, OADA, and the Trials Tracker App. 2018 ASABE Annual International Meeting. S.l.: American Society of Agricultural and Biological Engineers, pp. 1.

Obtener y filtrar datos de un dataframe. Disponible en: http://pyciencia.blogspot.com/2015/05/obtener-y-filtrar-datos-de-un-dataframe.html.

OPENSTREETMAP, A.E.M. y DE CARTOGRAFÍA, E.Y.D.W., 2012. TRABAJO FIN DE MÁSTER.

OTHMAN, M.S.B. y TAN, G., 2018. Machine Learning Aided Simulation of Public Transport Utilization. 2018 IEEE/ACM 22nd International Symposium on Distributed Simulation and Real Time Applications (DS-RT). S.l.: IEEE, pp. 1-2.

PARDO SIERRA, M.C., 2017. Nuevas herramientas para el análisis de la viabilidad de un modelo de negocio.

PARSONS, P.C., HUNG, Y.-H. y BAIGELENOV, A., 2018. Toward an Analysis of Practitioner-Oriented Resources for Visualization Design.

PÉREZ-MONTORO, M. Conceptos, contenidos y tecnología en visualización de información., vol. 13.

PETROU, T., 2017. Selecting Subsets of Data in Pandas : Part 1. Disponible en: https://medium.com/dunder-data/selecting-subsets-of-data-in-pandas-6fcd0170be9c.

PRUYT, E., ISLAM, T. y ARZT, T., 2018. On the Spot and Map: Interactive Model-Based Policy Support Under Deep Uncertainty. S.l.: Springer. Policy Analytics, Modelling, and Informatics.

Python data - Renombrando columnas en pandas. Disponible en : https://code.i-harness.com/es / q / ad216b.

Python pandas: compruebe si el valor es NaN en DataFrame. Disponible en: https://stackoverrun.com/es/q/8118136.

Python tutorial - seleccionar varias columnas por etiquetas pandas. Disponible en: https://code.i-harness.com/es / q / 1be31ec.

RODRÍGUEZ GONZÁLEZ, T., 2016. Espacios coworking.

ROVAI, M., 2018. Mapping Geograph Data in Python. Disponible en: https://towardsdatascience.com/mapping-geograph-data-in-python-610a963d2d7f.

SABIDÓ AGUADÉ, E., 2015. HTML5 application to locate nearby services using openstreetmap resources.

SARAJCEV, P. y MATIJASEVIC, H., 2018. Python in Lightning Detection Network Data Analysis. 25th International Lightning Detection Conference & 7th International Lightning Meteorology Conference. S.l.: s.n.

SEMLALI, B.B., EL AMRANI, C. y DENYS, S., 2019. Development of a Java-based application for environmental remote sensing data processing., vol. 9, no. 3, pp. 1978-1986.

SEVILLA-CALLEJO, M., 2017. OpenStreetMap.

TheSilphRoad - OSM Query for Map Features Used In Pokemon Go. Disponible en: https://www.reddit.com/r/TheSilphRoad/comments/6ujkkv/osm_query_for_map_features_used_in_pokemon_go/.

TheSilphRoad - OSM Query To Identify Possible Nests. Disponible en: https://www.reddit.com/r/TheSilphRoad/comments/54sy36/osm_query_to_identify_possible_nests/.

VAINGAST, S., 2014a. Beginning Python visualization: crafting visual transformation scripts. S.l. : Apress.

VAINGAST, S., 2014b. Graphs and Plots. S.l. : Springer. Beginning Python Visualization.

VASAVI, S., PRIYA, M.P. y GOKHALE, A.A., 2018. Framework for Geospatial Query Processing by Integrating Cassandra with Hadoop. S.l. : Springer. Knowledge Computing and Its Applications.

WOOD, M.A., 2015. Python and Matplotlib Essentials for Scientists and Engineers. S.l. : Morgan & Claypool Publishers.

YAP, R., 2018. A Simple Tutorial on How to document your Python Project using Sphinx and Rinohtype. Disponible en: https://medium.com/@richdayandnight/a-simple-tutorial-on-how-to-document-your-python-project-using-sphinx-and-rinohtype-177c22a15b5b.

YIM, A., CHUNG, C. y YU, A., 2018. Matplotlib for Python Developers: Effective techniques for data visualization with Python. S.l.: Packt Publishing Ltd.

YIN, D., LIU, Y., PADMANABHAN, A., TERSTRIEP, J., RUSH, J. y WANG, S., 2017. A CyberGIS-Jupyter framework for geospatial analytics at scale. Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact. S.l.: ACM, pp. 18.

ANEXO I - ABSTRACT (EN)

Python was introduced to the world of Geographic Information Systems (GIS) as a programming language relatively easy to learn and use. This language with the passage of time and its clear expansion, has become ubiquitous, offering solutions to different types of users.

This project merges two solutions that Python proposes, spatial analysis and the management of large volumes of data, so it can be used by companies using Big Data.

All this is done through the creation of a Python module that will combine it in a single block, transportable to any browser and to any mobile device. This module is completely based on several very important libraries in Python: Numpy, Pandas and Folium.

Leaflet is the main basis of this project, it is based on a JavaScript library that allows us to publish maps, statistics and graphics on the web quickly and efficiently. It has been possible to unite both worlds through Folium, manipulate data in Python and visualize large volumes of data for a GIS representation.

Visualize spatial data using maps has its advantages since you can obtain a visual representation of the exact location of the points, as a geographic data set, this allows us to easily relate the points we have with the real world. It also allows us to generate geographic perspectives from the data set and positions we have.

Plotting spatial data on a map allows us to obtain geographic information in a way that we cannot obtain with other forms of representation, or through other types of graphics.

Therefore, the use of maps instead of other forms of graphics allows us to highlight trends, statistics, discover patterns and reveal non-visible realities with representations of spatial data. Teaching us clarity about the data.

Since it is extremely difficult to obtain useful information from spatial data using tables as a form of representation, in this project these tasks have been combined, so that by means of tables (Pandas in Python) with a fairly large amount of data, we have been able to represent their values through spatial data on a map (Folium in Python).

During my university years, I have always been interested in research, and among all the fields covered during my career I have always been drawn to the field of Data Science.

In this field, when showing obtained results, its graphic representation is highly important, in order to be able to convey a simple way to understand the results of the study, whether intended for other researchers, colleagues or people interested in our publication.

The way of transmitting information graphically is very important, since it must be complemented in conjunction with our analytical results. By means of representations, patterns, differences, statistics, groupings or flows can be observed in a more visual and instantaneous way than if the results were represented in tables or in a static way. Always keeping in mind that this is not a way to simplify the result, but to highlight them and make them easier to observe and interpret.

Making a good graphical representation of data is not a simple problem, there are many studies on the subject that study human perception and how our brains interpret information, so it is always very important that the message is transmitted correctly.

At the beginning of this project, as in some work done during my studies, I observed that there is a common lack in all cases of visual representation of data. In most cases, representations are observed that do not reach the highest potential that could be acquired, if it is a good graphical representation; and in other cases, the representation is very vague or very unattractive, therefore, it does not attract attention.

There are countless libraries to do this graphical representation, but they are very complicated to use to get a good result, and usually require a lot of time. Therefore, my interest is to provide a tool that, in a clear and concise way, shows data scientists and researchers a good representation of the results, without requiring them to acquire high programming skills, and obtaining results in a short time. In this way, we can enhance its graphical representation and improve the transmission of information to readers.

During the career I have had several subjects related to programming, and especially with C, Java and Matlab. This has made me quite interested in programming focused on statistical studies and the more visual representation of data. Since I had some basis of programming, I decided to learn a new programming language, such is the case of Python, because thanks to it is having an explosion of its use along with the use of data in companies it has a great professional growth, and this seemed to me that it could also help me in the future.

Among the so-called “data science languages”, Python is one of the most widespread, commonly used to perform statistical calculation, as well as graphical representation of data, draw conclusions and be able to make final decisions of studies and research.

There is also a large community around, which provides tools and libraries for data mining, Big data, artificial intelligence and some other fields.

It is a very interesting language not only for visualizing data, but also for its ability to extract said data, to automate some processes or to use it in machine learning. With Python you can change, modify or extract large groups of interesting data through the use of libraries or modules, such as NumPy, Pandas, Folium, Json, ...

And although its use is destined for almost all the fields that encompass data science, the libraries destined to the graphical representation of data is one of the most important points. The number keeps growing, as each one is intended to adapt better to the input data they have, or to the type of output they want.

The Geographic Information System (GIS) is an integration manager that combines software with geographic data in order to design, capture, store, manipulate, analyze and deploy in any possible way, within its limitations, the geographically referenced information in order to solve more complex problems of management and planning.

Its main function is similar to that of a database containing geographic information (alphanumeric data), this information is associated through a common identifier to all the objects that belong to a digital map. By pointing to a single object all its attributes can be known, and by asking for one of the records in this database, its location within a set of geographical data can be obtained.

A GIS is mainly used to manage spatial information, as this system allows us to segment all the information belonging to the attributes of objects in different themed layers, and thus store them independently. These layers can be separated, allowing you to work with them easily and quickly, in order to get another new layer, which we could not generate otherwise.

The main issues that can be solved by a GIS are:

- Location: Allows to ask about the characteristics belonging to a specific place.
- Condition: The fulfillment of various conditions and functions imposed on the system.
- Trend: Comparisons between spatial and temporal situations different from any of the attributes.
- Routes: Calculation of optimal routes between points.
- Guidelines: Detection of specified spatial patterns.
- Models: The generation of models based on simulated phenomena or events.

As we can see, the field of application of GIS is very numerous, so it could be used in a multitude of activities, as long as it has a spatial component. It is the new technologies that have driven GIS to its decisive development, evolution and implementation.

Graphs and representations can give us a lot of information at a glance, but they are not always the best of representation methods. Sometimes it is necessary to go a little further and represent values, points of interest and statistics through a geographical map, which allows us to locate ourselves in the area and gives us the information we need. If the map also has the capacity to interact with us, so that we can travel with it and move to a different point to consult the same values and compare them, it would make it much easier for us to receive this information.

In order to meet the needs of many scientists and to be able to represent data along with geographical points and maps with Python, several libraries were born:

- PyNGL
- MATPLOTLIB
- BASEMAP
- MPLOT3D
- SEABORN
- BOKEH
- PyGAL
- IPYLEAFLET
- FOLIUM

Since the objective of this project is to elaborate a good geographical representation, we have isolated in the search for the best bookshop two candidates to develop with them all the proposed research. These two finalists are Ipyleaflet and Folium.

At first, it was very expensive to find a suitable library in the ever-changing world of Python libraries, but after some research I discovered Folium, which facilitates the creation and manipulation of geographical data representations using Python.

To conclude the use of Folium, we have thoroughly tested the mentioned libraries, carefully analyzing their advantages and disadvantages. And after comparing the results we have observed that the most suitable for our purpose are only two: Ipyleaflet and Folium.

Since our goal is not to limit the use of the library that we implement only to the scope of Jupyter, and despite that, over a period of time, we began to implement in Ipyleaflet.

Finally, it was concluded to develop our module using Folium for the representation of geographic data and maps.

The functions to be implemented must be able to represent using the Folium library as a base:

1. Points of interest in a geographical representation (map) from a Data frame. Points of interest that are indicated in a Data frame file will be represented in a simple and fast way.
2. Choropleth maps from a Data frame. Maps containing statistical information, which can be displayed with color scales, e.g. a map painting population density in countries will be automatically represented.
3. Points of interest from URL links. Points of interest will be automatically represented on a map knowing only a URL that provides information about that point.
4. Points of interest from Geohash coordinates, located in a geographical representation starting from a Data frame containing Geohash information, translating into latitude-longitude coordinates.
5. Density areas from latitude-longitude coordinates. So that larger circles or icons are shown if there are several nearby points that might overlap on the map. This representation is open to other options that allow to represent the same objective.

The overall objective of the module is the implementation of high-level interfaces for the graphical representation of geospatial data contained in Data frames. An intuitive and interactive representation of a dataset is sought.

The programming language for its implementation is Python and the data input format will be modular and customizable, we will be able to introduce data, mainly through Data frames from Pandas library (for example, CSV or JSON format).

The implemented functions should take advantage of the low-level libraries already available. The objective is to promote these libraries so that they adapt to our use cases.

The case of use that motivates the fulfillment of this bachelor thesis, is the lack of these interfaces when analyzing online advertising data with geographical targeting of users. For this reason, the library will at least have functions that allow it to be used in this context but bearing in mind that they can be easily applied for other different analyses with input data in the same format.

In the section on building resources, we have used the Overpass API.

A great tool for mapping, as it is very powerful for filtering OSM data. Through the Overpass turbo mode, there is an easy way to quickly execute any Overpass query and inspect the results in a very simple way for the user.

Overpass turbo is mainly a useful tool for the general public, to filter certain things you are looking for.

But there are several ideas for which to use Overpass turbo, so that it is a quite useful tool for mapping:

- Find special Points of interest (POI) that are not drawn on the map.
- Check POIs that are evenly distributed over large areas.
- Show large spatial features (parks, rivers, boundaries, complete motorways, service stations, tourist areas...) and load them directly into an editor.
- Use only a filtered portion of the OSM data.

For developers, Overpass Turbo offers some options:

Permite probar y desarrollar consultas de API Overpass más o menos complicadas.

- Allows you testing and developing more or less complicated Overpass API queries.
- Converts OSM data to geoJSON data format.
- Creates clickable or static map layouts that highlight selected OSM features.

To query in Overpass turbo, place the Overpass API query in the editor, press the Run button and wait for the OSM dataset to appear, sometimes this wait is very long, you can modify the waiting time in milliseconds in the code that appears on the left. It has a query wizard, which is designed to transform simple, user-readable search terms into functional Overpass queries. Alternatively, it lets you write a label, to perform the query.

Overpass turbo displays as much data as possible and when you click on an object a pop-up window displays all the information for the selected point. Also, different types of identification, labels, coordinates and metadata, if available for our consultation.

It is a quite complete tool, which we will use for the POI search that will be returned in geoJSON data format.

To use it, you need a recent and updated web browser, we have tested Opera, Chrome and Firefox, with all of them it works correctly.

With respect to future lines of work, we propose options that have emerged after learning to use Python libraries, such as Folium or Pandas and after understanding what a Data frame is, and where to get them (Overpass Turbo). After knowing how to work with them, we have seen that due to lack of time and level of development knowledge we have not been able to implement all the functions we would have liked.

During the development of this project we have been opening a lot of possibilities and many ideas where to apply everything learned, to enhance our module.

- One of the most interesting options that we have observed, and that due to lack of time in the development we have not been able to implement, is the idea of using image recognition algorithms to create a function, which receiving as input an image (a photograph taken with a drone) may be able to determine and represent the point at which we are on a map.

  For example:

  If with a drone we took a photograph of the inner courtyard of the Carlos III University and passed the image captured by our algorithm, the image would be recognized, contrasting the colors with the Google Maps knowledge database, and all its topographic information. In this way, it could be determined that by means of a concordance of more than 98%, he could tell us exactly what point we are photographing and from there he would begin to explore the map from that point, with this procedure we could even trace a route from the University to our home.

- Another line that can be followed in the future is the improvement of something that already exists in our module, specifically we would improve the function that uses the URLs to add to the points of interest (POI) contrasted information of the site to represent.

  We would have liked to add to the commercial POIs the business information of this item.

For example:

If the point of interest were a restaurant, we would add information about the type of food that the restaurant offers.

We are aware that this is already implemented in other applications in the market, but the functionality we want to expand is to be able to save our own POI map and feedback it to have a knowledge database fed.

In short, we wanted to simplify a whole world. But concluding that a person who is dedicated to analyze data can see it without the need to program, since it would already be programmed/done. And available to open as a web page in any modern and updated browser.

For example, for an agency, you could show a list of sites where customers buy the most and depending on where you are most likely to sell there, that is where you would invest.

It can also be very useful when you need to give an explanation to the client, or to elaborate a presentation in which you want to show statistics, in a visual way. So that. without previous knowledge of Python, nor of programming directly, the desired data can be shown graphically and visually.

Key words:

Choropleth maps; GeoHashes; Folium; Python's library; Pandas.

ANEXO II - README

https://github.com/IsabelaSJ/data2maps

Trabajo final de grado de Isabela San José García por la UC3M, Diseño e implementación de un módulo Python para representar datos geográficos en Dataframes

Resumen

Este módulo trata de una librería Python la cual de manera sencilla es capaz de transformar un conjunto grande de datos de entrada en representaciones geográficas de dichos datos. Esta librería engloba unas funciones concretas que tienen la capacidad de ser adaptables a la representación de estadísticas y estudios geográficos. Las funciones de las que consta la librería descrita son:

- Representaciones de Dataframes como puntos de interés.
- Representación de Dataframes en mapas coropléticos.
- Representación de Dataframes con urls de Google Maps.
- Representaciones de Dataframes en Geohash.

Modo de uso

Instalar el paquete mediante pip:

```
pip install data2maps
```

Para usarlo dentro de un script de Python o en un módulo:

```
import data2maps
```

Te dará acceso a las siguientes funciones:

- poi
- choropleth
- gmaps_url
- geohash

En la carpeta `./examples/` se encuentran cuadernos de Jupyter con ejemplos detallados del uso de cada una de las funciones.

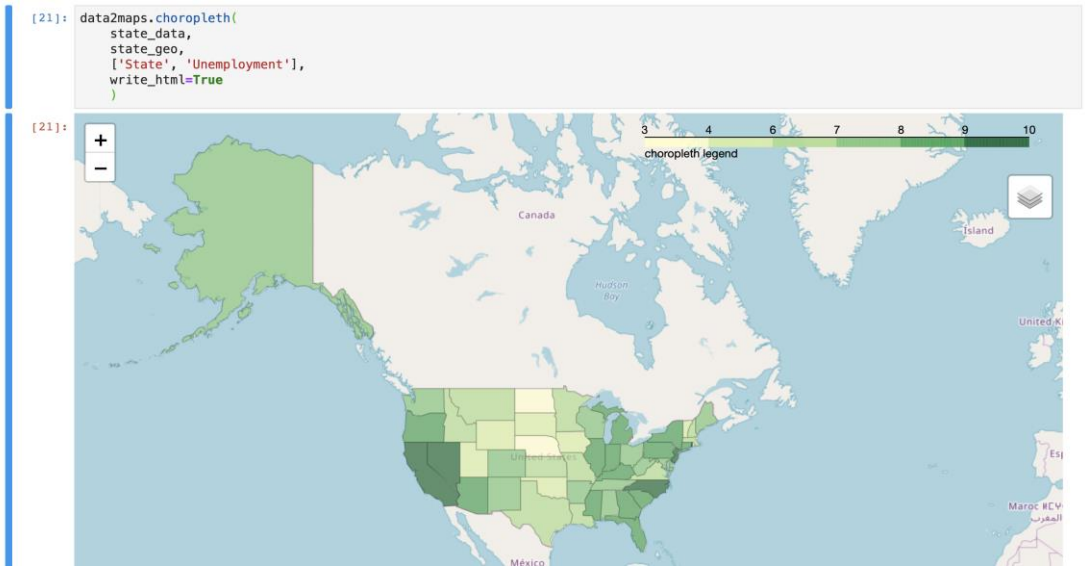

Ilustración 32. Anexo II. Ejemplo de uso de la función choropleth.

Para usarlo por comandos:

```
data2maps --type poi --input data.csv --out poi.html
```

- –type: Parámetro obligatorio. Puede tomar los valores de [poi | choropleth | gmaps_url | geohash]
- –input: Parámetro obligatorio. Indica la ruta del fichero de entrada
- –out: Parámetro obligatorio. Indica el nombre del fichero de salida
- –color: Valor opcional. Indica la tonalidad de salida del mapa
- –clustered: Valor opcional. Agrupa los puntos cercanos, solo válido para los mapas poi y gmaps_url
- –columns: Valor opcional. Indica el nombre de las columnas desde donde se leen las coordenadas y descripciones
- –leyend: Indica el nombre de las leyendas

Desarrollo

Dependencias

- PyBuilder - Sirve para construir el proyecto, con los siguientes plugins:
  - unittest - Para ejecutar test unitarios
  - mockito - Utilidad de mock para los tests

  - coverage - Informe de covertura de tests
  - distutils - Compilación del proyecto
  - flake8 - Linter con ayudas de código para evitar errores en tiempo de ejecución

- Sphinx - Generar documentación a partir del código y comentarios
- Pandas
- Folium
- GoogleMaps
- re - Expresiones regulares
- geohash2
- geopy - Utilidad de mapas y cálculo de distancias
- twine - Publicar compilados en PyPI

Instalación de entorno de desarrollo

Para clonar el proyecto

```
git clone https://github.com/IsabelaSJ/data2maps
```

Para crear virtualenv de Python3

```
pip3 install virtualenv
virtualenv -p Python3 venv
source venv/bin/activate
```

Instalar recursivamente las dependencias de Python que se encuentran en el fichero requirements.txt con su versión correspondiente:

```
pip install -r requirements.txt
```

Instalar los plugins de pybuilder, están en el fichero build.py, contenidos en la función use_plugin([NOMBRE_PLUGIN]):

```
pyb install_dependencies
```

Generar la documentación, usamos el plugin de pybuilder llamado ‘Sphinx’, que lee todo el Python doc y genera un html en la carpeta ./doc/_build:

```
pyb sphinx_generate_documentation
```

Para compilar el código:

```
pyb –verbose
```

Actualizar versión

Cuando modificamos el código para actualizar la versión debemos hacerlo en los siguientes ficheros:

- build.py
- ./docs/conf.py

Y volvemos a compilar el código:

```
pyb --verbose
```

Esto nos generará una nueva carpeta en ./target/dist/data2maps-[NÚMERO_VERSIÓN] con los ficheros del paquete listos para subir al repositorio de Python PyPI

Subir/Actualizar package en PyPI

Crear fichero de credenciales para el repositorio de paquetes PyPI.

~/.pypirc

```
[distutils]
index-servers=
pypitest
pypi
[pypitest]
repository = https://upload.pypi.org/legacy/
username = [NOMBRE_USUARIO]
password = [CONTRASEÑA]

[pypi]
repository = https://upload.pypi.org/legacy/
username = [NOMBRE_USUARIO]
password = [CONTRASEÑA]
```

Enlace con credenciales a dos repositorios de código:

- **pypi:** Repositorio oficial de pip, una vez subido a este repositorio estará accesible a todo el mundo para instalar mediante el comando pip
- **pypitest:** Repositorio para pruebas de pip, que sirve para probar el módulo antes de subirlo al repositorio público, esto evita ensuciarlo con códigos inacabados o que no estén del todo depurados.

Para subir el código ejecutamos el siguiente comando:

```
twine upload -r pypi target/dist/data2maps-1.0/dist/* --verbose
```